Descriptive complexity of real computation and probabilistic independence logic

Hannula, Miika

IEEE Computer Society
2020-07

Hannula , M , Kontinen , J , Bussche , J V D & Virtema , J 2020 , Descriptive complexity of real computation and probabilistic independence logic . in Proceedigs of the Thirty-Fifth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) . IEEE Computer þÿ S o c i e t y , p p . 5 5 0 5 6 3 , T h i r t y - F i f t h A n n u a l A C M / I E E E S y m p o s i u m o n L o Science (LICS) , 08/07/2020 . https://doi.org/10.1145/3373718.3394773

http://hdl.handle.net/10138/319065
https://doi.org/10.1145/3373718.3394773

acceptedVersion

Descriptive complexity of real computation and probabilistic independence logic

Miika Hannula
University of Helsinki
Finland
miika.hannula@helsinki.fi

Juha Kontinen
University of Helsinki
Finland
juha.kontinen@helsinki.fi

Jan Van den Bussche
Hasselt University
Belgium
jan.vandenbussche@uhasselt.be

Jonni Virtema
Hokkaido University
Japan
jonni.virtema@let.hokudai.ac.jp
Hasselt University
Belgium

Abstract

We introduce a novel variant of BSS machines called Separate Branching BSS machines (S-BSS in short) and develop a Fagin-type logical characterisation for languages decidable in nondeterministic polynomial time by S-BSS machines. We show that NP on S-BSS machines is strictly included in NP on BSS machines and that every NP language on S-BSS machines is a countable union of closed sets in the usual topology of $\mathbb{R}^n$. Moreover, we establish that on Boolean inputs NP on S-BSS machines without real constants characterises a natural fragment of the complexity class $\exists\mathbb{R}$ (a class of problems polynomial time reducible to the true existential theory of the reals) and hence lies between NP and PSPACE. Finally we apply our results to determine the data complexity of probabilistic independence logic.

CCS Concepts: • **Theory of computation → Complexity theory and logic**; **Finite Model Theory**; **Models of computation**; • **Mathematics of computing → Probability and statistics**.

Keywords: Blum-Shub-Smale machines, descriptive complexity, team semantics, independence logic, real arithmetic.

ACM Reference Format:
Miika Hannula, Juha Kontinen, Jan Van den Bussche, and Jonni Virtema. 2020. Descriptive complexity of real computation and probabilistic independence logic. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '20), July 8–11, 2020, Saarbrücken, Germany.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3373718.3394773

LICS '20, July 8–11, 2020, Saarbrücken, Germany

ACM ISBN 978-1-4503-7104-9/20/07…$15.00
https://doi.org/10.1145/3373718.3394773

1 Introduction

The existential theory of the reals consists of all first-order sentences that are true about the reals and are of the form

$$\exists x_1 \dots \exists x_n \phi(x_1, \dots, x_n),$$

where ϕ is a quantifier-free arithmetic formula containing inequalities and equalities. Known to be NP-hard on the one hand, and in PSPACE on the other hand [6], the exact complexity of this theory is a major open question. The existential theory of the reals is today attracting considerable interest due to its central role in geometric graph theory. First isolated as a complexity class in its own right in [25], $\exists\mathbb{R}$ is defined as the closure of the existential theory of the reals under polynomial-time reductions. In the past decade several algebraic and geometric problems have been classified as complete for $\exists\mathbb{R}$; a recent example is the art gallery problem of deciding whether a polygon can be guarded by a given number of guards [1].

The existential theory of the reals is closely connected to *Blum-Shub-Smale machines* (BSS machine for short) which are essentially random access machines with registers that can store arbitrary real numbers and which can compute rational functions over reals in a single time step. Many complexity classes from classical complexity theory transfer to the realm of BSS machines, such as nondeterministic polynomial time ($\mathsf{NP}_\mathbb{R}$) over languages consisting of finite strings of reals. While the focus is primarily on languages over some numerical domain (e.g., reals or complex numbers), also Boolean inputs (strings over $\{0, 1\}$) can be considered. In this context $\exists\mathbb{R}$ corresponds to the *Boolean part* of $\mathsf{NP}^0_\mathbb{R}$ ($\mathrm{BP}(\mathsf{NP}^0_\mathbb{R})$), obtained by restricting $\mathsf{NP}_\mathbb{R}$ to Boolean inputs and limiting the use of machine constants to 0 and 1, as feasibility of Boolean

combinations of polynomial equations is complete for both of these classes [5, 26].

BSS computations can also be described logically. This research orientation was initiated by Grädel and Meer who showed that $\mathsf{NP}_{\mathbb{R}}$ is captured by a variant of existential second-order logic ($\mathrm{ESO}_{\mathbb{R}}$) over *metafinite structures* [15]. Metafinite structures are two-sorted structures that consist of a finite structure, an infinite domain with some arithmetics (such as the reals with multiplication and addition), and weight functions bridging the two sorts [13]. Since the work by Grädel and Meer, others (see, e.g., [8, 18, 24]) have shed more light upon *the descriptive complexity over the reals* mirroring the development of classical descriptive complexity. In addition to metafinite structures, the connection between logical definability encompassing numerical structures and computational complexity has received attention in *constraint databases* [2, 14, 23]. A constraint database models, e.g., geometric data by combining a numerical *context structure*, such as the real arithmetic, with a finite set of quantifier-free formulae defining infinite database relations [20].

In this paper we investigate the descriptive complexity of so-called *probabilistic independence logic* in terms of the BSS model of computation and the existential theory of the reals. Probabilistic independence logic is a recent addition to the vast family of new logics in *team semantics*. In team semantics [27] formulae are evaluated with respect to sets of assignments which are called teams. During the past decade research on team semantics has flourished with interesting connections to fields such as database theory [17], statistics [7], hyperproperties [22], and quantum information theory [19], just to mention a few examples. The focus of this article is probabilistic team semantics that extends team based logics with probabilistic dependency notions. While the first ideas of probabilistic teams trace back to [11, 19], the systematic study of the topic was initiated by the works [9, 10].

At the core of probabilistic independence logic $\mathrm{FO}(\perp\!\!\!\perp_c)$ is the concept of conditional independence. The models of this logic are finite first-order structures but the notion of a team is replaced by a probabilistic team, i.e., a discrete probability distribution over a finite set of assignments. In [10] it was observed that probabilistic independence logic is equivalent to a restriction of $\mathrm{ESO}_{\mathbb{R}}$ in which the weight functions are distributions. The exact complexity and relationship of $\mathrm{FO}(\perp\!\!\!\perp_c)$ to $\mathrm{ESO}_{\mathbb{R}}$ and $\mathsf{NP}_{\mathbb{R}}$ was left as an open question; in this paper we present a (strict) sublogic of $\mathrm{ESO}_{\mathbb{R}}$ and a (strict) subclass of $\mathsf{NP}_{\mathbb{R}}$ that both capture $\mathrm{FO}(\perp\!\!\!\perp_c)$.

Our contribution. In this paper we introduce a novel variant of BSS machines called Separate Branching BSS machines (S-BSS machines for short) and characterise its NP languages (denoted by $\text{S-NP}_{[0,1]}$) with $\text{L-ESO}_{[0,1]}[+,\times,\leq,(r)_{r\in\mathbb{R}}]$ that is a natural sublogic of $\mathrm{ESO}_{\mathbb{R}}$. Likewise, we isolate a fragment $\exists[0,1]^{\leq}$ of the complexity class $\exists\mathbb{R}$ and show that it coincides with the class of Boolean languages in $\text{S-NP}^0_{[0,1]}$.

$$\begin{array}{ccccccc} & & \mathrm{BP}(\text{S-NP}^0_{[0,1]}) & & \mathrm{BP}(\mathsf{NP}^0_{\mathbb{R}}) & & \\ \mathsf{NP} & \subseteq & \| ^* & \subseteq & \| & \subseteq & \mathsf{PSPACE} \\ & & \exists[0,1]^{\leq} & & \exists\mathbb{R} & & \end{array}$$

$$\begin{array}{ccc} \text{S-NP}^0_{[0,1]} & & \mathsf{NP}^0_{\mathbb{R}} \\ \| ^* & \subset^* & \| \\ \text{L-ESO}_{[0,1]}[+,\times,\leq,0,1] & & \mathrm{ESO}_{\mathbb{R}}[+,\times,\leq,0,1] \\ \| ^* & & \\ \mathrm{FO}(\perp\!\!\!\perp_c) & & \end{array}$$

Table 1. Known complexity results and logical characterisations together with the main results of this paper. The results of this paper are marked with an asterisk (*). The top figure is with respect to Boolean inputs; on the bottom figure, the inputs can include real numbers.

Moreover we establish a topological characterisation of the languages decidable by S-BSS machines; we show that, under certain natural restrictions, languages decidable by S-BSS machines are disjoint unions of closed sets in the usual topology of $\mathbb{R}^n$. The topological characterisation separates the languages decidable by BSS machines and S-BSS machines, respectively. Moreover it enables us to separate the complexity classes $\text{S-NP}^0_{[0,1]}$ and $\mathsf{NP}^0_{\mathbb{R}}$. Finally we show the equivalence of the logics $\text{L-ESO}_{[0,1]}[+,\times,\leq,0,1]$ and $\mathrm{FO}(\perp\!\!\!\perp_c)$, implying that $\mathrm{FO}(\perp\!\!\!\perp_c) \equiv \text{S-NP}^0_{[0,1]}$. Table 1 summarises the main results of the paper.

Structure of the paper. In Section 2 we give the basic definitions related to descriptive complexity, BSS machines, and logics on $\mathbb{R}$-structures required for this paper. Section 3 focuses in giving logical characterisations of variants of NP on S-BSS machines. In Section 4 we establish the aforementioned topological characterisation of S-BSS decidable languages. In Section 5 we prove a hierarchy of the related complexity classes; in particular we separate $\text{S-NP}^0_{[0,1]}$ and $\mathsf{NP}^0_{\mathbb{R}}$. Section 6 deals with probabilistic team semantics and establishes that $\mathrm{FO}(\perp\!\!\!\perp_c) \equiv \text{S-NP}^0_{[0,1]}$. Section 7 concludes the paper.

2 Preliminaries

A vocabulary is *relational* (resp., *functional*) if it consists of only relation (resp., function) symbols. A structure is *relational* if it is defined over a relational vocabulary. We let Var_1 and Var_2 denote disjoint countable sets of first-order and function variables (with prescribed arities), respectively. We write $\vec{x}$ to denote a tuple of first-order variables and $|\vec{x}|$ to denote the length of that tuple. The arities of function variables f and relation symbols R are denoted by $\mathrm{ar}(f)$ and $\mathrm{ar}(R)$, respectively. If f is a function with domain $\mathrm{Dom}(f)$ and A a set, we define $f \restriction A$ to be the function with domain $\mathrm{Dom}(f) \cap A$ that agrees with f for each element in its domain. Given a finite set D, a function $f\colon D \to [0,1]$ that maps elements of D to elements of the closed interval $[0,1]$ of real numbers such that $\sum_{s\in D} f(s) = 1$ is called a *(probability) distribution*.

2.1 $\mathbb{R}$-structures

Let τ be a relational vocabulary. A τ-structure is a tuple $\mathfrak{A} = (A, (R^{\mathfrak{A}})_{R\in\tau})$, where A is a nonempty set and each $R^{\mathfrak{A}}$ an $\mathrm{ar}(R)$-ary relation on A. The structure $\mathfrak{A}$ is *a finite structure* if τ and A are finite sets. In this paper, we consider structures that enrich finite relational τ-structures by adding real numbers ($\mathbb{R}$) as a second domain sort and functions that map tuples over A to $\mathbb{R}$.

Definition 2.1. Let τ and σ be respectively a finite relational and a finite functional vocabulary, and let $X \subseteq \mathbb{R}$. An X-structure of vocabulary $\tau \cup \sigma$ is a tuple

$$\mathfrak{A} = (A, \mathbb{R}, (R^{\mathfrak{A}})_{R\in\tau}, (g^{\mathfrak{A}})_{g\in\sigma}),$$

where the reduct of $\mathfrak{A}$ to τ is a finite relational structure, and each $g^{\mathfrak{A}}$ is a *weight function* from $A^{\mathrm{ar}(g)}$ to X. Additionally, an $d[0,1]$-structure $\mathfrak{A}$ is defined analogously, with the exception that the weight functions $g^{\mathfrak{A}}$ are distributions.

An assignment is a total function $s : \mathrm{Var}_1 \to A$ that assigns a value for each first-order variable. The modified assignment $s[a/x]$ is an assignment that maps x to a and agrees with s for all other variables.

Next, we define a variant of functional existential second-order logic with numerical terms ($\mathrm{ESO}_{\mathbb{R}}$) that is designed to describe properties of $\mathbb{R}$-structures. As first-order terms we have only first-order variables. For a set σ of function symbols, the set of numerical σ-terms i is generated by the following grammar:

$$i ::= c \mid f(\vec{x}) \mid i \times i \mid i + i \mid \mathrm{SUM}_{\vec{y}}\, i,$$

where $c \in \mathbb{R}$ is a real constant denoting itself, $f \in \sigma$, and $\vec{x}$ and $\vec{y}$ are tuples of first-order variables from Var_1 such that the length of $\vec{x}$ is $\mathrm{ar}(f)$. The value of a numerical term i in a structure $\mathfrak{A}$ under an assignment s is denoted by $[i]^{\mathfrak{A}}_s$. In addition to the natural semantics for the real constants, we have the following rules for the numerical terms:

$$[f(\vec{x})]^{\mathfrak{A}}_s := f^{\mathfrak{A}}(s(\vec{x})), \qquad [i \times j]^{\mathfrak{A}}_s := [i]^{\mathfrak{A}}_s \cdot [j]^{\mathfrak{A}}_s,$$
$$[i + j]^{\mathfrak{A}} := [i]^{\mathfrak{A}} + [j]^{\mathfrak{A}}, \qquad [\mathrm{SUM}_{\vec{y}}\, i]^{\mathfrak{A}}_s := \sum_{\vec{a}\in A^{|\vec{y}|}} [i]^{\mathfrak{A}}_{s[\vec{a}/\vec{y}]},$$

where $+, \cdot, \sum$ are the addition, multiplication, and summation of real numbers, respectively.

Definition 2.2 (Syntax of $\mathrm{ESO}_{\mathbb{R}}$)**.** Let τ be a finite relational vocabulary and σ a finite functional vocabulary. Let $O \subseteq \{+, \times, \mathrm{SUM}\}$, $E \subseteq \{=, <, \leq\}$, and $C \subseteq \mathbb{R}$. The set of $\tau \cup \sigma$-formulae of $\mathrm{ESO}_{\mathbb{R}}[O, E, C]$ is defined via the grammar:

$$\phi ::= x = y \mid \neg x = y \mid i\ e\ j \mid \neg i\ e\ j \mid R(\vec{x}) \mid \neg R(\vec{x}) \mid$$
$$\phi \wedge \phi \mid \phi \vee \phi \mid \exists x\phi \mid \forall x\phi \mid \exists f\psi,$$

where i and j are numerical σ-terms constructed using operations from O and constants from C, and $e \in E$, $R \in \tau$ is a relation symbol, f is a function variable, x and y are first-order variables and $\vec{x}$ a tuple of first-order variables, and ψ is a $\tau \cup (\sigma \cup \{f\})$-formula of $\mathrm{ESO}_{\mathbb{R}}[O, E, C]$.

Note that the syntax of $\mathrm{ESO}_{\mathbb{R}}[O, E, C]$ allows first-order subformulae to appear only in negation normal form. This restriction however does not restrict the expressiveness of the language.

The semantics of $\mathrm{ESO}_{\mathbb{R}}[O, E, C]$ is defined via $\mathbb{R}$-structures and assignments analogous to first-order logic; note that first-order variables are always assigned to a value in A whereas functions map tuples over A to $\mathbb{R}$. In addition to the clauses of first-order logic, we have the following semantical clauses:

$$\mathfrak{A} \models_s i\ e\ j \Leftrightarrow [i]^{\mathfrak{A}}_s\ e\ [j]^{\mathfrak{A}}_s, \quad \mathfrak{A} \models_s \neg i\ e\ j \Leftrightarrow \mathfrak{A} \not\models_s i\ e\ j,$$
$$\mathfrak{A} \models_s \exists f\phi \Leftrightarrow \mathfrak{A}[h/f] \models_s \phi \text{ for some } h\colon A^{\mathrm{ar}(f)} \to \mathbb{R}, \quad (1)$$

where $\mathfrak{A}[h/f]$ denotes the expansion of $\mathfrak{A}$ that interprets f as h.

Given $S \subseteq \mathbb{R}$, we define $\mathrm{ESO}_S[O, E, C]$ as the variant of $\mathrm{ESO}_{\mathbb{R}}[O, E, C]$ in which (1) is modified such that $h\colon A^{\mathrm{ar}(f)} \to S$, and $\mathrm{ESO}_{d[0,1]}[O, E, C]$ as the variant in which (1) is modified such that $h\colon A^{\mathrm{ar}(f)} \to [0, 1]$ is a distribution, that is, $\Sigma_{\vec{a}\in A^{\mathrm{ar}(f)}} h(\vec{a}) = 1$. Note that in the setting of $\mathrm{ESO}_{d[0,1]}[O, E, C]$ the value $f^{\mathfrak{A}}$ of a 0-ary function symbol f is always 1.

Loose fragment. For both $S \subseteq \mathbb{R}$ and $S = d[0, 1]$, define L-$\mathrm{ESO}_S[O, E, C]$ as the *loose fragment* of $\mathrm{ESO}_S[O, E, C]$ in which negated numerical atoms $\neg i\ e\ j$ are disallowed. We want to point out that as long as $= \in E$ and $0, 1 \in C$, the logic L-$\mathrm{ESO}_S[O, E, C]$ subsumes existential second-order logic over finite structures (a precise formulation is given later by Proposition 3.1).

Expressivity comparisons. Fix a relational vocabulary τ and a functional vocabulary σ. Let $\mathcal{L}$ and $\mathcal{L}'$ be some logics over $\tau \cup \sigma$ defined above, and let $X \subseteq \mathbb{R}$ or $X = d[0, 1]$. For a formula $\phi \in \mathcal{L}$, define $\mathrm{Struc}^X(\phi)$ to be the class of X-structures $\mathfrak{A}$ of vocabulary $\tau \cup \sigma$ such that $\mathfrak{A} \models \phi$. We write $\mathcal{L} \leq_X \mathcal{L}'$ if for all sentences $\phi \in \mathcal{L}$ there is a sentence $\psi \in \mathcal{L}'$ such that $\mathrm{Struc}^X(\phi) = \mathrm{Struc}^X(\psi)$. As usual, the shorthand $\equiv_X$ stands for $\leq_X$ in both directions. For $X = \mathbb{R}$, we write simply $\leq$ and $\equiv$.

In plain words, the subscript S in $\mathrm{ESO}_S[O, E, C]$ constitutes the class of functions available for quantification, whereas the superscript X in $\mathrm{Struc}^X(\phi)$ constitutes the class of functions available for function symbols in the vocabulary. Thus, $\phi \in \mathrm{ESO}_S[O, E, C]$ defines a class $\mathrm{Struc}^X(\phi)$, even if S and X are different.

2.2 Blum-Shub-Smale Model

We will next give a definition of BSS machines (see e.g. [3]). We define $\mathbb{R}^* := \bigcup\{\mathbb{R}^n \mid n \in \mathbb{N}\}$. The *size* $|x|$ of $x \in \mathbb{R}^n$ is defined as n. The space $\mathbb{R}^*$ can be seen as the real analogue of Σ^* for a finite set Σ. We also define $\mathbb{R}_*$ as the set of all sequences $x = (x_i)_{i\in\mathbb{Z}}$ where $x_i \in \mathbb{R}$. The members of $\mathbb{R}_*$ are thus of the form $(\ldots, x_{-2}, x_{-1}, x_0, x_1, x_2, \ldots)$. Given an element $x \in \mathbb{R}^* \cup \mathbb{R}_*$ we write x_i for the ith coordinate of x. The space $\mathbb{R}_*$ has natural shift operations. We define shift left

$\sigma_l : \mathbb{R}_* \to \mathbb{R}_*$ and shift right $\sigma_r : \mathbb{R}_* \to \mathbb{R}_*$ as $\sigma_l(x)_i := x_{i+1}$ and $\sigma_r(x)_i := x_{i-1}$.

Definition 2.3 (BSS machines)**.** A BSS machine consists of an input space $\mathcal{I} = \mathbb{R}^*$, a state space $\mathcal{S} = \mathbb{R}_*$, and an output space $\mathcal{O} = \mathbb{R}^*$, together with a connected directed graph whose nodes are labelled by $1, \ldots, N$. The nodes are of five different types.

1. *Input node*. The node labeled by 1 is the only input node. The node is associated with a next node $\beta(1)$ and the input mapping $g_I : \mathcal{I} \to \mathcal{S}$.
2. *Output node*. The node labeled by N is the only output node. This node is not associated with any next node. Once this node is reached, the computation halts, and the result of the computation is placed on the output space by the output mapping $g_O : \mathcal{S} \to \mathcal{O}$.
3. *Computation nodes.* A computation node m is associated with a next node $\beta(m)$ and a mapping $g_m : \mathcal{S} \to \mathcal{S}$ such that for some $c \in \mathbb{R}$ and $i, j, k \in \mathbb{Z}$ the mapping g_m is identity on coordinates $l \neq i$ and on coordinate i one of the following holds:
   - $g_m(x)_i = x_j + x_k$ (addition),
   - $g_m(x)_i = x_j - x_k$ (subtraction),
   - $g_m(x)_i = x_j \times x_k$ (multiplication),
   - $g_m(x)_i = c$ (constant assignment).
4. *Branch nodes.* A branch node m is associated with nodes $\beta^-(m)$ and $\beta^+(m)$. Given $x \in \mathcal{S}$ the next node is $\beta^-(m)$ if $x_0 \leq 0$, and $\beta^+(m)$ otherwise.
5. *Shift nodes.* A shift node m is associated either with shift left σ_l or shift right σ_r, and a next node $\beta(m)$.

The input mapping $g_I : \mathcal{I} \to \mathcal{S}$ places an input $(x_1, \ldots, x_n)$ in the state

$$(\ldots, 0, n, x_1, \ldots, x_n, 0, \ldots) \in \mathcal{S},$$

where the size of the input n is located at the zeroth coordinate. The output mapping $g_O : \mathcal{S} \to \mathcal{O}$ maps a state to the string consisting of its first l positive coordinates, where l is the number of consecutive ones stored in the negative coordinates starting from the first negative coordinate. For instance, g_O maps

$$(\ldots, 2, 1, 1, 1, n, x_1, x_2, x_3, x_4, \ldots) \in \mathcal{S},$$

to $(x_1, x_2, x_3) \in \mathcal{O}$. A configuration at any moment of computation consists of a node $m \in \{1, \ldots, N\}$ and a current state $x \in \mathcal{S}$. The (sometimes partial) *input-output* function $f_M : \mathbb{R}^* \to \mathbb{R}^*$ of a machine M is now defined in the obvious manner. A function $f : \mathbb{R}^* \to \mathbb{R}^*$ is *computable* if $f = f_M$ for some machine M. A language $L \subseteq \mathbb{R}^*$ is *decided* by a BSS machine M if its characteristic function $\chi_L : \mathbb{R}^* \to \mathbb{R}^*$ is f_M.

Deterministic complexity classes. A machine M *runs in (deterministic) time* $t : \mathbb{N} \to \mathbb{N}$, if M reaches the output in $t(|x|)$ steps for each input $x \in \mathcal{I}$. The machine M runs in *polynomial time* if t is some polynomial function. The complexity class $\mathsf{P}_\mathbb{R}$ is defined as the set of all subsets of $\mathbb{R}^*$ that are decided by some machine M running in polynomial time.

Nondeterministic complexity classes. A language $L \subseteq \mathbb{R}^*$ is *decided nondeterministically* by a BSS machine M, if

$$x \in L \quad \text{if and only if} \quad f_M((x, x')) = 1, \text{ for some } x' \in \mathbb{R}^*,$$

when a slightly different input mapping $g_I : \mathcal{I} \to \mathcal{S}$, which places an input $(x_1, \ldots, x_n, x'_1, \ldots, x'_m)$ in the state

$$(\ldots, 0, n, m, x_1, \ldots, x_n, x'_1, \ldots, x'_m, \ldots) \in \mathcal{S},$$

where the sizes of x and x' are respectively placed on the first two coordinates, is used. When we consider languages that a machine M decides nondeterministically, we simply call M *nondeterministic*. Sometimes when we wish to emphasize that this is not the case, we call M *deterministic*. Moreover, we say that M is *[0,1]-nondeterministic*, if the guessed strings x' are required to be from $[0, 1]^*$. L is *decided in time* $t : \mathbb{N} \to \mathbb{N}$, if, for every $x \in L$, M reaches the output 1 in $t(|x|)$ steps for some $x' \in \mathbb{R}^*$. The machine *runs in polynomial time* if t is a polynomial function. The class $\mathsf{NP}_\mathbb{R}$ consists of those languages $L \subseteq \mathbb{R}^*$ for which there exists a machine M that nondeterministically decides L in polynomial time. Note that, in this case, the size of x' above can be bounded by a polynomial (e.g., the running time of M) without altering the definition. The complexity class $\mathsf{NP}_\mathbb{R}$ has many natural complete problems such as 4-FEAS, i.e., the problem of determining whether a polynomial of degree four has a real root [4].

Complexity classes with Boolean restrictions. If we restrict attention to machines M that may use only $c \in \{0, 1\}$ in constant assignment nodes, then the corresponding complexity classes are denoted using an additional superscript 0 (e.g., as in $\mathsf{NP}^0_\mathbb{R}$). Complexity classes over real computation can also be related to standard complexity classes. For a complexity class C over the reals, the *Boolean part* of C, written $\mathrm{BP}(C)$, is defined as $\{L \cap \{0, 1\}^* \mid L \in C\}$.

Descriptive complexity. Similar to Turing machines, also BSS machines can be studied from the vantage point of descriptive complexity. To this end, finite $\mathbb{R}$-structures are encoded as finite strings of reals using so-called rankings that stipulate an ordering on the finite domain. Let $\mathfrak{A}$ be an $\mathbb{R}$-structure over $\tau \cup \sigma$ where τ and σ are relational and functional vocabularies, respectively. A *ranking* of $\mathfrak{A}$ is any bijection $\pi : \mathrm{Dom}(A) \to \{1, \ldots, |A|\}$. A ranking π and the lexicographic ordering on $\mathbb{N}^k$ induce a *k-ranking* $\pi_k : \mathrm{Dom}(A)^k \to \{1, \ldots, |A|^k\}$ for $k \in \mathbb{N}$. Furthermore, π induces the following encoding $\mathrm{enc}_\pi(\mathfrak{A})$. First we define $\mathrm{enc}_\pi(R^\mathfrak{A})$ and $\mathrm{enc}_\pi(f^\mathfrak{A})$ for $R \in \tau$ and $f \in \sigma$:

- Let $R \in \tau$ be a k-ary relation symbol. The encoding $\mathrm{enc}_\pi(R^\mathfrak{A})$ is a binary string of length $|A|^k$ such that the jth symbol in $\mathrm{enc}_\pi(R^\mathfrak{A})$ is 1 if and only if $(a_1, \ldots, a_k) \in R^\mathfrak{A}$, where $\pi_k(a_1, \ldots, a_k) = j$.
- Let $f \in \sigma$ be a k-ary function symbol. The encoding $\mathrm{enc}_\pi(f^\mathfrak{A})$ is string of real numbers of length $|A|^k$ such that the jth symbol in $\mathrm{enc}_\pi(f^\mathfrak{A})$ is $f^\mathfrak{A}(\vec{a})$, where $\pi_k(\vec{a}) = j$.

The encoding $\mathrm{enc}_\pi(\mathfrak{A})$ is then the concatenation of the string $(1,\dots,1)$ of length $|A|$ and the encodings of the interpretations of the relation and function symbols in $\tau\cup\sigma$. We denote by $\mathrm{enc}(\mathfrak{A})$ any encoding $\mathrm{enc}_\pi(\mathfrak{A})$ of $\mathfrak{A}$.

Let C be a complexity class and $\mathrm{ESO}_S[O,E,C]$ a logic, where $O\subseteq\{+,\times,\mathrm{SUM}\}$, $E\subseteq\{=,<,\le\}$, $C\subseteq\mathbb{R}$, and $S\subseteq\mathbb{R}$ or $S=d[0,1]$. Let $X\subseteq\mathbb{R}$ or $X=d[0,1]$, and let $\mathcal{S}$ be an arbitrary class of X-structures over $\tau\cup\sigma$ that is closed under isomorphisms. We write $\mathrm{enc}(\mathcal{S})$ for the set of encodings of structures in $\mathcal{S}$. Consider the following two conditions:

(i) $\mathrm{enc}(\mathcal{S})=\{\mathrm{enc}(\mathfrak{A})\mid\mathfrak{A}\in\mathrm{Struc}^X(\phi)\}$ for some $\phi\in\mathrm{ESO}_S[O,E,C][\tau\cup\sigma]\}$,
(ii) $\mathrm{enc}(\mathcal{S})\in C$.

If (i) implies (ii), we write $\mathrm{ESO}_S[O,E,C]\le_X C$, and if the vice versa holds, we write $C\le_X\mathrm{ESO}_S[O,E,C]$. If both directions hold, then we write $\mathrm{ESO}_S[O,E,C]\equiv_X C$. We omit the subscript X in the notation if $X=\mathbb{R}$.

The following results due to Grädel and Meer extends Fagin's theorem to the context of real computation.[1]

Theorem 2.4 ([15]). $\mathrm{ESO}_{\mathbb{R}}[+,\times,\le,(r)_{r\in\mathbb{R}}]\equiv\mathsf{NP}_{\mathbb{R}}$ *and* $\mathrm{ESO}_{\mathbb{R}}[+,\times,\le]\equiv\mathsf{NP}^0_{\mathbb{R}}$.

2.3 Separate Branching BSS

We now define a restricted version of the BSS model which branches with respect to two separated intervals $(-\infty,\epsilon^-]$ and $[\epsilon^+,\infty)$. We will later relate these BSS machines to certain fragments of $\mathrm{ESO}_{\mathbb{R}}$ and the existential theory of the reals.

Definition 2.5 (Separate Branching BSS Machine)**.** *Separate branching BSS machines* (S-BSS machines for short) are otherwise identical to the BSS machines of Definition 2.3, except that the branch nodes are replaced with the following *separate branch nodes*.

- *Separate branch nodes.* A separate branch node m is associated with $\epsilon_-,\epsilon_+\in\mathbb{R}$, $\epsilon_-<\epsilon_+$, and nodes $\beta^+(m)$ and $\beta^-(m)$. Given $x\in\mathcal{S}$ the next node is $\beta^+(m)$ if $x_0\ge\epsilon_+$, $\beta^-(m)$ if $x_0\le\epsilon_-$, and otherwise the input is rejected.

Note that for a given S-BSS machine it is easy to write an equivalent BSS machine. A priori it is not clear whether the converse is possible; in fact, we will later show that in some cases the converse is not possible.

We can now define the variants of the complexity classes $\mathsf{P}_{\mathbb{R}}$, $\mathsf{P}^0_{\mathbb{R}}$, $\mathsf{NP}_{\mathbb{R}}$, and $\mathsf{NP}^0_{\mathbb{R}}$ that are obtained by replacing BSS machines with S-BSS machines in the definitions of the complexity classes. Furthermore, we define $\mathsf{NP}_{[0,1]}$, and $\mathsf{NP}^0_{[0,1]}$ as the variants of $\mathsf{NP}_{\mathbb{R}}$, and $\mathsf{NP}^0_{\mathbb{R}}$ in which the input x may be any element from $\mathbb{R}^*$ but the guessed element x' must be taken from $[0,1]^*$. Let C be one of the aforementioned complexity classes. We define S-C to be the variant of C, where, instead of BSS machines, S-BSS machines are used. If C includes the superscript 0, this means that not only the parameter c in constant assignment, but also ϵ_- and ϵ_+ in separate branching are from $\{0,1\}$.

[1] Only the first equivalence is explicitly stated in [15]. The second, however, is a simple corollary, using the fact that 0 and 1 can be identified in $\mathrm{ESO}_{\mathbb{R}}[+,\times,\le]$; these two are the only idempotent reals for multiplication, and 0 is the only idempotent real for addition.

3 Descriptive complexity of nondeterministic polynomial time in S-BSS

We now show that $\text{S-}\mathsf{NP}_{[0,1]}$ corresponds to a numerical variant of ESO in which quantified functions may only take values from the unit interval and numerical inequality atoms may only appear positively. Later we will show that both of these restrictions are necessary in the sense that removing either one lifts expressiveness to the level of $\mathrm{ESO}_{\mathbb{R}}[+,\times,\le,(r)_{r\in\mathbb{R}}]$ which captures $\mathsf{NP}_{\mathbb{R}}$. On the other hand, we give a logical proof, based on topological arguments, that $\text{S-}\mathsf{NP}_{[0,1]}<\mathsf{NP}_{\mathbb{R}}$.

The proof of Theorem 3.3 is a nontrivial adaptation of the proof of Theorem 2.4 (see [15, Theorem 4.2]). In the proof we apply Lemma 3.2 and, by Proposition 3.1, assume without loss of generality built-in ESO definable predicates on the finite part.

Let 0 and 1 be two distinct constants, d be a $(k+1)$-ary distribution, and R a k-ary relation on a finite domain A of size n. We say that d is the *characteristic distribution* of R (w.r.t. 0 and 1) if $\vec{a}\in R$ implies $d(\vec{a},1)=\frac{1}{n^k}$, and $\vec{a}\notin R$ implies $d(\vec{a},0)=\frac{1}{n^k}$. The next proposition implies that it is possible to simulate existential quantification of ESO definable predicates on the finite domain using function (or distribution) quantification; in particular, we may assume without loss of generality built-in predicates such as a linear ordering and its induced successor relation on the finite domain. Clearly, any predicate that is ESO-definable over finite structures is also ESO-definable (with respect to the finite domain) over $\mathbb{R}$-structures.

Below, we write $\text{L-ESO}_S[O,E,C,\exists X]$ to denote the extension of $\text{L-ESO}_S[O,E,C]$ by existential quantification of relations over the finite domain with the usual semantics.

Proposition 3.1. *Let* $\{0,1\}\subseteq S$ *and* O,E,C *be arbitrary. For every formula* $\phi\in\text{L-ESO}_S[O,E,C,\exists X]$ *there exist formulas* $\phi'\in\text{L-ESO}_S[O,E\cup\{=\},C\cup\{0,1\}]$ *and* $\phi''\in\text{L-ESO}_{d[0,1]}[O,E\cup\{=\},C]$ *such that*

$$\mathfrak{A}\models_s\phi\ \Leftrightarrow\ \mathfrak{A}\models_s\phi'\ \Leftrightarrow\ \mathfrak{A}\models_s\phi'',$$

for every $\mathbb{R}$*-structure* $\mathfrak{A}$ *and assignment* s.

Proof. The sentence ϕ' (ϕ'', resp.) is obtained from ϕ by a recursive translation that is the identity for all other cases, except that, for second-order variables X of arity k, we rewrite the quantifications $\exists X$ as $\exists f_X$, where f_X is an k-ary ($(k+1)$-ary, resp.) function variable, and the atoms $X(\vec{x})$ and $\neg X(\vec{x})$ by $f_X(\vec{x})=1$ and $f_X(\vec{x})=0$ ($f_X(\vec{x},1)=u(\vec{x})$ and $f_X(\vec{x},0)=u(\vec{x})$, resp.), respectively. Here, u is the k-ary uniform distribution which is definable in the logic $\text{L-ESO}_{d[0,1]}[=]$ by $\forall\vec{x}\vec{x}'u(\vec{x})=u(\vec{x}')$. □

Lemma 3.2. *If* $\{0,1\} \subseteq C$*, we have* $\text{L-ESO}_{[0,1]}[+,\times,\leq,C] \equiv \text{L-ESO}_{[-1,1]}[+,\times,\leq,C]$.

Proof. Left-to-right direction is straightforward; the quantification $\exists f\, \psi$ in $\text{L-ESO}_{[0,1]}[+,\times,\leq,C]$ can be simulated in $\text{L-ESO}_{[-1,1]}[+,\times,\leq,C]$ by the formula

$$\exists f(\forall \vec{x}\, 0 \leq f(\vec{x}) \wedge \psi).$$

The converse direction is nontrivial. Let ϕ be an arbitrary $\text{L-ESO}_{[-1,1]}[+,\times,\leq,C]$-formula. We will show how to construct an equivalent $\text{L-ESO}_{[0,1]}[+,\times,\leq,C]$-formula ϕ'. By the standard Skolemization argument we may assume that ϕ is in the prenex normal form. Moreover, we assume that every atomic formula of the form $t_1 \leq t_2$ is written such that t_1 and t_2 are multivariate polynomials where function terms $f(\vec{x})$ play the role of variables; this normal form is obtained by using the distributive laws of addition and multiplication. Let M be the smallest set that includes every term of polynomials t_1 and t_2 such that $t_1 \leq t_2$ is a subformula of ϕ, and is closed under taking subterms. Clearly M is a finite set, for its cardinality is bounded by the length of ϕ. For each $p \in M$ with m variables, we introduce an m-ary function g_p that will be interpreted as the sign function for the term p. Let $\vec{x}_p$ be the related tuple of variables. The idea is that $g_p(\vec{a}) = 0$ ($g_p(\vec{a}) = 1$) if $p(\vec{a}) < 0$ ($p(\vec{a}) \geq 0$).

We are now ready to define the translation $\phi \mapsto \phi'$, where

$$\phi = \exists f_1 \ldots \exists f_m Q_1 x_1 \ldots Q_n x_n\, \psi$$

is in the normal form mentioned above. We define

$$\phi' := \exists_{p \in M} g_p \exists f_1 \ldots \exists f_m Q_1 x_1 \ldots Q_n x_n (\theta \wedge \psi^\circ),$$

where the recursively defined translation $^\circ$ is homomorphic for the Boolean connectives and identity for first-order literals.

For atomic formulae $t_1 \leq t_2$ of the form $s_1 + \cdots + s_l \leq r_1 + \cdots + r_m$ the translation is defined as follows. The translation makes certain that every term (of polynomial) of the inequation after the translation has a non-negative value; this is done by moving terms to the other side of the inequation. Denote $\mathcal{I} = \{1,\ldots,l\}$ and $\mathcal{J} = \{1,\ldots,m\}$, and define $(t_1 \leq t_2)^\circ$ as

$$\bigvee_{\substack{I \subseteq \mathcal{I} \\ J \subseteq \mathcal{J}}} \Big(\bigwedge_{\substack{i \in I \\ j \in J}} g_{s_i}(\vec{x}_{s_i}) = 1 \wedge g_{r_j}(\vec{x}_{r_j}) = 1 \wedge \bigwedge_{\substack{i \in \mathcal{I} \setminus I \\ j \in \mathcal{J} \setminus J}} g_{s_i}(\vec{x}_{s_i}) = 0 \wedge g_{r_j}(\vec{x}_{r_j}) = 0 \wedge \sum_{i \in I} s_i + \sum_{j \in \mathcal{J} \setminus J} r_j \leq \sum_{i \in \mathcal{I} \setminus I} s_i + \sum_{j \in J} r_j \Big).$$

Finally the subformula θ makes sure that the signs of the terms in $p \in M$ propagate correctly from subterms to terms. Define θ as

$$\bigwedge_{\substack{p \in M \\ c \in M \cap [0,\infty] \\ d \in M \cap [-\infty,0)}} \forall \vec{x} \big(g_p(\vec{x}) = 0 \vee g_p(\vec{x}) = 1 \big) \wedge g_c = 1 \wedge g_d = 0$$

$$\wedge \bigwedge_{\substack{p,q,r \in M \\ p = q \times r}} \Big(\big(g_q(\vec{x}_q) = g_r(\vec{x}_r) \wedge g_p(\vec{x}_p) = 1 \big) \vee \big(g_q(\vec{x}_q) = 0 \wedge g_r(\vec{x}_r) = 1 \wedge g_p(\vec{x}_p) = 0 \big) \vee \big(g_q(\vec{x}_q) = 1 \wedge g_r(\vec{x}_r) = 0 \wedge g_p(\vec{x}_p) = 0 \big) \Big).$$

Note that the sign function maps terms of value 0 to either 0 or 1, since for the purpose of the construction the sign of 0 valued terms does not matter. □

Theorem 3.3. $\text{L-ESO}_{[0,1]}[+,\times,\leq,(r)_{r\in\mathbb{R}}] \equiv \text{S-NP}_{[0,1]}$.

Proof. ***Right-to-left direction.*** Suppose $L \in \text{S-NP}_{[0,1]}$ is a class of $\mathbb{R}$-structures that is closed under isomorphisms. By Lemma 3.2 it suffices to construct an $\text{L-ESO}_{[-1,1]}[+,\times,\leq,\mathbb{R}]$ sentence ϕ such that $\mathfrak{A} \models \phi$ iff $\mathfrak{A} \in L$ for all $\mathbb{R}$-structures $\mathfrak{A}$. Let M be an S-BSS machine such that M consists of N nodes, and for each input x it accepts (x, x') for some $x' \in [0,1]^*$ in time $|x|^{k^*}$ iff $x = \text{enc}(\mathfrak{A})$ for some $\mathfrak{A} \in L$, where k^* is some fixed natural number. We may assume that $|x'|$ is of size $|x|^{k^*}$. Let k be a fixed natural number such that $|x|^{k^*} \leq |A|^k$; such a k always exists since $|\text{enc}(\mathfrak{A})|$ is polynomial in $|A|$. The computation of M on a given input $\text{enc}(\mathfrak{A})$ can be represented using functions $f : A^{2k+1} \to (-1,1)$, $g : A^{2k+1} \to (0,1]$, and $h_1,\ldots,h_N : A^k \to \{0,1\}$ such that

(a) $f(\vec{s},\vec{t})/g(\vec{s},\vec{t})$ is the content of register $\vec{s}$ at time $\vec{t}$;
(b) $h_i(\vec{t})$ is 1 if i is the node label at time $\vec{t}$, and 0 otherwise.

Note that $\vec{s}$ is $(k+1)$-ary because we need to store $|A|^k$ positive and negative register contents. We may assume k such that registers with index greater than $|A|^k$ do not contribute to the final outcome, i.e., their contents are never shifted to registers associated with the nodes of M. Construct a formula

$$\psi(f,g,h) := \theta_{\text{pre}} \wedge \theta_{\text{initial}} \wedge \theta_{\text{comp}} \wedge \theta_{\text{accept}}$$

of $\text{L-ESO}_{[-1,1]}[+,\times,\leq,(r)_{r\in\mathbb{R}}]$ such that $\mathfrak{A} \models \exists f g h\, \psi$ iff M accepts $\text{enc}(\mathfrak{A})$. By Proposition 3.1 we may assume a built-in ordering $\leq_{\text{fin}}$ and its induced successor relation S and constants 0, 1, max on the finite domain. Likewise, we may extend $\leq_{\text{fin}}$ to order also k-tuples from the finite domain. Under such ordering we then write $\vec{x}+1$ ($\vec{x}-1$) for the element succeeding (preceding) a k-tuple $\vec{x}$, and $\vec{n}$ for the n-th k-tuple. First, θ_{pre} is the conjunction of a formula stating that the ranges of g and h are as stated, and another formula

$$\forall \vec{y}\, f(\vec{y})^2 + g(\vec{y}) = 1, \tag{2}$$

where $f(\vec{y})^2$ is a shorthand for $f(\vec{y}) \times f(\vec{y})$. Observe that (2) implies

$$\frac{f(\vec{y})}{g(\vec{y})} = \frac{f(\vec{y})}{(1 - f(\vec{y})^2)}.$$

Also, $x \mapsto x/(1-x^2)$ is a bijection from $(-1,1)$ to $\mathbb{R}$. That the range of f is $(-1,1)$ will follow from the remaining conjuncts of ψ, described below.

Initial configuration. We give a description of θ_{initial} such that

$$(\mathfrak{A}, f, g, \vec{h}) \models \theta_{\text{initial}} \quad \text{iff } (f, g, \vec{h}) \text{ satisfies (a) \& (b) at time } \vec{0}. \tag{3}$$

For clause (b) it suffices to add to θ_{initial}

$$h_1(\vec{0}) = 1 \wedge h_2(\vec{0}) = 0 \wedge \ldots \wedge h_N(\vec{0}) = 0.$$

Consider then clause (a). We denote by $\vec{s}_0$ the $\lfloor |A^{k+1}|/2 \rfloor$th $k+1$-tuple with respect to $\leq_{\text{fin}}$. The sequence $\vec{s}_0$, which is clearly definable in ESO, now represents the zeroth coordinate of R_*. To encode that $|x|$ is placed on zeroth coordinate we add to θ_{initial}

$$\begin{aligned}\exists \epsilon \exists f_{\text{count}} \Big(& f_{\text{count}}(0) = \epsilon \\ & \wedge \forall xy \big(S(x,y) \rightarrow f_{\text{count}}(y) = f_{\text{count}}(x) + \epsilon \big) \\ & \wedge f_{\text{count}}(\max) = 1 \wedge f(\vec{s}_0, \vec{0}) = p(1/\epsilon) \times g(\vec{s}_0, \vec{0}) \Big),\end{aligned} \tag{4}$$

where ϵ is a nullary function variable (i.e., a real from $[-1,1]$), p is a polynomial such that $|\text{enc}(\mathfrak{A})| = p(|A|)$, and the last conjunct of (4) is a shorthand for

$$\epsilon^{\deg(p)} \times f(\vec{s}_0, \vec{0}) = p^*(\epsilon) \times g(\vec{s}_0, \vec{0}),$$

where $\deg(p)$ is the degree of the polynomial p, and p^* is the polynomial obtained by multiplying p by $\epsilon^{\deg(p)}$ (that is $\epsilon^{\deg(p)} \times p(1/\epsilon) = p^*(\epsilon)$). It follows from (2) and (4) that $f(\vec{s}_0, \vec{0}) \in (-1,1)$ and $f(\vec{s}_0, \vec{0})/g(\vec{s}_0, \vec{0}) = |\text{enc}(\mathfrak{A})|$. To encode that $|x'|$ is placed on the first coordinate we also add to θ_{initial} a formula stipulating that $f(\vec{s}_0, \vec{0})^{k^*}/g(\vec{s}_0, \vec{0})^{k^*} = f(\vec{s}_0 + 1, \vec{0})/g(\vec{s}_0 + 1, \vec{0})$.

Let $f^* \in \tau$ be a function symbol and let r_{f^*} be a natural number that indicates the starting position of the encoding of f^* in $\text{enc}(\mathfrak{A})$. Clearly r_{f^*} is a definable real number as it is the value of a fixed univariate polynomial. We use the shorthand $\vec{s} = \vec{y} + r_{f^*}$ to denote that in the ordering of k-tuples (induced from $\leq_{\text{fin}}$) the ordinal number of $\vec{s}$ is the sum of the ordinal number of $\vec{y}$ and r_{f^*}. Clearly $\vec{s} = \vec{y} + r_{f^*}$ is expressible in our logic. We then add the following to θ_{initial}:

$$\forall \vec{s} \vec{y} \bigwedge_{f^* \in \tau} \Big(\vec{s} = \vec{y} + r_{f^*} \rightarrow \big(f(\vec{s}, \vec{0}) = f^*(\vec{y}) \times g(\vec{s}, \vec{0}) \big) \Big) \tag{5}$$

Note that (2) and (5) imply that $f(\vec{s}, \vec{0}) \in (-1,1)$; for, by (2), $|f(s,0)| = 1$ leads to $g(s,0) = 0$ which contradicts (5). The interpretations of relations in σ are treated analogously. For all the remaining positions $\vec{s} > \vec{s}_0$ we stipulate that $0 \leq f(\vec{s}, \vec{0}) \leq g(\vec{s}, \vec{0})$, and for all positions $\vec{s} < \vec{s}_0$ we stipulate that $f(\vec{s}, \vec{0}) = 0$. In the first case $f(\vec{s}, \vec{0})/g(\vec{s}, \vec{0})$ is some value guessed from the unit interval $[0,1]$ and in the second case it is 0. We conclude that (3) holds by this construction.

Computation configurations. Then we define θ_{comp} such that

$$(\mathfrak{A}, f, g, \vec{h}) \models \theta_{\text{comp}} \quad \text{iff } (f, g, \vec{h}) \text{ satisfies (a) and (b) at time } \vec{t} > \vec{0}. \tag{6}$$

We let

$$\begin{aligned}\theta_{\text{comp}} := \forall \vec{s} \vec{t} \Big(& \bigvee_{1 \leq m < m' \leq N} \big(h_m(\vec{t}) = 0 \vee h_{m'}(\vec{t}) = 0 \big) \wedge \\ & \bigvee_{1 \leq m \leq N} \big(h_m(\vec{t}) = 1 \wedge \theta_m \big) \Big),\end{aligned}$$

where each θ_m describes the instruction of node m. Suppose m is a computation node associated with a mapping g_m that is the identity on coordinates $l \neq i$ and on coordinate i defined as $g_m(x)_i = x_j + x_k$. Let us write $f_{\vec{s},\vec{t}}$ and $g_{\vec{s},\vec{t}}$ for $f(\vec{s},\vec{t})$ and $g(\vec{s},\vec{t})$, and $\vec{s}_i, \vec{s}_j, \vec{s}_k$ for the tuples that correspond to the ith, jth, and kth input coordinates. Clearly, these tuples are definable. We define

$$\begin{aligned}\theta_m := & h_{\beta(m)}(\vec{t}+1) = 1 \wedge f_{\vec{s}_i,\vec{t}+1} \times g_{\vec{s}_j,\vec{t}} \times g_{\vec{s}_k,\vec{t}} \\ & = g_{\vec{s}_i,\vec{t}+1} \times (f_{\vec{s}_j,\vec{t}} \times g_{\vec{s}_k,\vec{t}} + g_{\vec{s}_j,\vec{t}} \times f_{\vec{s}_k,\vec{t}}) \wedge \\ & \vec{s} \neq \vec{s}_i \rightarrow (f_{\vec{s},\vec{t}+1} = f_{\vec{s},\vec{t}} \wedge g_{\vec{s},\vec{t}+1} = g_{\vec{s},\vec{t}}).\end{aligned}$$

The other computation nodes are described analogously. For a shift left node m we define

$$\begin{aligned}\theta_m := & h_{\beta(m)}(\vec{t}+1) = 1 \wedge \\ & \vec{s} < \vec{\max} \rightarrow (f_{\vec{s},\vec{t}+1} = f_{\vec{s}+1,\vec{t}} \wedge g_{\vec{s},\vec{t}+1} = g_{\vec{s}+1,\vec{t}}),\end{aligned}$$

and the case for shift right node is analogous. For a separate branch node m we define

$$\begin{aligned}\theta_m := & \Big(\big(h_{\beta^+(m)}(\vec{t}+1) = 1 \wedge f_{\vec{s}_0,\vec{t}} \geq \epsilon^+ \big) \vee \\ & \big(h_{\beta^-(m)}(\vec{t}+1) = 1 \wedge f_{\vec{s}_0,\vec{t}} \leq \epsilon^- \big) \Big) \wedge \\ & f_{\vec{s},\vec{t}+1} = f_{\vec{s},\vec{t}} \wedge g_{\vec{s},\vec{t}+1} = g_{\vec{s},\vec{t}}.\end{aligned}$$

Our formulae now imply that (6) follows by the construction. In particular, keeping the values of f in $(-1,1)$ ensures that the arithmetical operations are encoded correctly.

Finally, to express that the value of the characteristic function f_M is 1 we may stipulate without loss of generality that coordinates $-2, -1, 1$ respectively contain $0, 1, 1$; we also need to state that the machine is in node N at the last step:

$$\begin{aligned}\theta_{\text{accept}} := & h_N(\vec{\max}) = 1 \wedge f_{\vec{s}_0+1,\vec{\max}} = g_{\vec{s}_0+1,\vec{\max}} \\ & \wedge f_{\vec{s}_0-1,\vec{\max}} = g_{\vec{s}_0-1,\vec{\max}} \wedge f_{\vec{s}_0-2,\vec{\max}} = 0.\end{aligned}$$

We conclude that $\mathfrak{A} \models \exists f g \vec{h}\, \psi$ iff M accepts $\text{enc}(\mathfrak{A})$.

Left-to-right direction. Let $\phi \in \text{L-ESO}_{[0,1]}[+, \times, \leq, \mathbb{R}]$ be a sentence over some vocabulary $\sigma \cup \tau$. As in the previous lemma, we may assume that ϕ is of the form

$$\exists f_1 \ldots \exists f_m Q_1 x_1 \ldots Q_n x_n\, \psi,$$

where ψ is quantifier-free. We may further may transform ϕ to an equivalent form

$$\exists f_1 \ldots \exists f_m \exists g_{i_{l+1}} \ldots \exists g_{i_n} \forall x_{i_1} \ldots \forall x_{i_l}\, \psi', \tag{7}$$

where g_{i_j} are Skolem functions on the finite domain and ψ' is obtained from ψ by replacing each occurrence of x_{i_j}, $l+1 \leq j \leq n$, with $g_{i_j}(\vec{x}_j)$. Note that (7) is an intermediate expression which is not anymore in L-ESO$_{[0,1]}[+,\times,\leq,\mathbb{R}]$. We may assume ψ' is in disjunctive normal form $\bigvee_{i\in I} C_i$, where I is a finite set of indices.

Suppose the relational and function symbols in $\sigma \cup \tau \cup \{f_1,\ldots,f_m\}$ are of arity at most $n' \geq n$. First, a fixed initial segment of negative coordinates is allocated with the following intention:

- one coordinate a for separate branching,
- three coordinates i,j,k for numerical identity atoms,
- two sequences of coordinates $\vec{b} = (b_1,\ldots,b_n)$ and $\vec{c} = (c_1,\ldots,c_{n'})$ for elements of the finite domain.

We construct a machine M which runs in polynomial time and accepts (x,x') iff

1. $x = \mathrm{enc}(\mathfrak{A})$ where $\mathfrak{A}$ is a model over $\sigma\cup\tau$, and
2. (x,x') is a concatenation of $\mathrm{enc}((\mathfrak{A},\vec{f},\vec{g}))$ and indices $i_{\vec{a}} \in I$ such that $(\mathfrak{A},\vec{f},\vec{g},\vec{a}) \models C_{i_{\vec{a}}}$ for each $\vec{a}\in A^l$.

We may suppose that $\vec{f}$ and $(\vec{g},(i_{\vec{a}})_{\vec{a}\in A^l})$ are respectively encoded as strings of reals and integers.

Let p' be a polynomial such that for each $\mathfrak{A}$ over $\sigma\cup\tau$ we have $p'(|A|) = \mathrm{enc}(\mathfrak{A})$. The machine first checks whether there is a natural number d such that $p'(d) = |x|$. For this, it first sets $x_i \leftarrow 1$ and $x_a \leftarrow x_0 - p'(x_i)$, where initially $x_0 = |x|$. If $x_a = 0$, then $x_0 \leftarrow x_i$, and if $x_a \geq 1$, then $x_i \leftarrow x_i + 1$ and the process is repeated. Otherwise, if $x_a \notin \{0\}\cup[1,\infty)$, the input is rejected. This type of branching can be implemented repeating separate branching twice. Provided that the input is not rejected, this process terminates with $x_0 = d$ where $p'(d) = |x|$. The machine then checks whether item 1 holds; given $|\mathfrak{A}|$ this is straightforward. Checking that (x,x') is a concatenation of $\mathrm{enc}((\mathfrak{A},\vec{f},\vec{g}))$, for some functions $\vec{f},\vec{g}$, and some indices $i_{\vec{a}}$ is analogous.

It remains to be checked that the last claim of item 2 holds. We go through all tuples $\vec{a}\in A^l$, calculate the values of the Skolem functions, and check that the disjunct $C_{i_{\vec{a}}}$ holds for the calculated value of the variables. For each $\vec{a} = (a_1,\ldots,a_l) \in \{0,\ldots,d-1\}^l$, placed on the coordinates $b_1,\ldots,b_l$, the machine uses x_0 and $\vec{c}$ for retrieving and placing $g_{i_{l+1}}(\vec{a}_{l+1}),\ldots,g_{i_n}(\vec{a}_n)$ on the coordinates $b_{l+1},\ldots,b_n$. The machine then retrieves the index $i_{\vec{a}}$ and checks whether $C_{i_{\vec{a}}}$ holds true with respect to the values on coordinates $\vec{b}$. Once this process is completed for all value combinations $(a_1,\ldots,a_l)\in\{0,\ldots,d-1\}^l$ the computation halts with accept.

The contents of the input are accessed using shifts which fix the contents of the allocated coordinates. That is, we use operations σ_l^X, where X is a finite set of coordinates, such that $\sigma_l^X(x)_i = x_i$ if $i\in X$, and otherwise $\sigma_l^X(x)_i = x_j$ where $j = \min\{k\in\mathbb{N} \mid k > i, k\notin X\}$. For instance, $\sigma_l^{\{0\}}$ is obtained by first swapping x_0 and x_1 and then shifting left.

Also, if $C_{i_{\vec{a}}}$ contains a numerical atom $f(\vec{t}_0) \leq g(\vec{t}_1)\times h(\vec{t}_2)$, then the values of its constituent function terms with respect to $\vec{b}$ are placed on coordinates i,j,k. The machine then sets $x_a \leftarrow x_i - x_j \times x_k$, and if $x_a \leq 0$, then it continues to the next atom in $C_{i_{\vec{a}}}$, and else it rejects. If $C_{i_{\vec{a}}}$ contains a relational atom $R(\vec{x}_0)$, then the value of its characteristic function with respect to $\vec{b}$ is placed on coordinate a. If $x_a = 1$, then the machine moves to the next atom in $C_{i_{\vec{a}}}$, and else it rejects. Negated relational atoms are treated analogously, and the stated branching is straightforward to implement with separate branch nodes.

It follows from our construction that M runs in polynomial time and accepts (x,x') iff items 1 and 2 hold. Hence, we conclude that L-ESO$_{[0,1]}[+,\times,\leq,(r)_{r\in\mathbb{R}}] \leq$ S-NP$_{[0,1]}$. □

Suppose we above consider (i) guesses from $\mathbb{R}$ instead of $[0,1]$, or (ii) BSS instead of S-BSS machines. Then slightly modified proofs yield (i) L-ESO$_{\mathbb{R}}[+,\times,\leq,(r)_{r\in\mathbb{R}}] \equiv$ S-NP$_{\mathbb{R}}$, and (ii) ESO$_{[0,1]}[+,\times,\leq,(r)_{r\in\mathbb{R}}] \equiv$ NP$_{[0,1]}$. Furthermore, logical constants $r\in\mathbb{R}\setminus\{0,1\}$ are only needed to capture c in constant assignment and ϵ^+,ϵ^- in separate branching, and for the converse direction only those machine constants $r\in\mathbb{R}\setminus\{0,1\}$ which explicitly occur in the logical expression are needed. Thus we obtain the following corollary.

Corollary 3.4.

1. L-ESO$_{\mathbb{R}}[+,\times,\leq,(r)_{r\in\mathbb{R}}] \equiv$ S-NP$_{\mathbb{R}}$,
2. L-ESO$_{\mathbb{R}}[+,\times,\leq,0,1] \equiv$ S-NP$^0_{\mathbb{R}}$,
3. L-ESO$_{[0,1]}[+,\times,\leq,0,1] \equiv$ S-NP$^0_{[0,1]}$,
4. ESO$_{[0,1]}[+,\times,\leq,(r)_{r\in\mathbb{R}}] \equiv$ NP$_{[0,1]}$,
5. ESO$_{[0,1]}[+,\times,\leq,0,1] \equiv$ NP$^0_{[0,1]}$.

In the following two sections we investigate how S-BSS computability relates to BSS computability, and in particular how S-NP$_{[0,1]}$ relates to NP$_{\mathbb{R}}$. On the one hand it turns out that S-NP$_{[0,1]}$ is strictly weaker than NP$_{\mathbb{R}}$. On the other hand both obvious strengthenings of S-NP$_{[0,1]}$, namely S-NP$_{\mathbb{R}}$ and NP$_{[0,1]}$, collapse to NP$_{\mathbb{R}}$.

4 Characterisation of S-BSS decidable languages

We give a characterisation of languages decidable by S-BSS machines using the ideas from the previous section. The goal of this section is to establish the following theorem:

Theorem 4.1. *Every language that can be decided by a) a deterministic S-BSS machine, or b) a $[0,1]$-nondeterministic S-BSS machine in time t, for some Turing computable function $t\colon\mathbb{N}\to\mathbb{N}$, is a countable disjoint union of closed sets in the usual topology of $\mathbb{R}^n$.*

The result complements an analogous characterisation of BSS-decidable languages thus giving insight on the difference of the computational powers of BSS machines and S-BSS machines.

Theorem 4.2 ([3, Theorem 1]). *Every language decidable by a (deterministic) BSS machine is a countable disjoint union of semi-algebraic sets.*

These characterisations are based on the fact that the computation of BSS and S-BSS machines can be encoded by formulae of first-order real arithmetic.

Existential theory of the real arithmetic. Formulae of the *existential real arithmetic* are given by the grammar

$$\phi ::= i \leq i \mid i < i \mid \phi \wedge \phi \mid \phi \vee \phi \mid \exists x \phi, \tag{8}$$

where i stands for numerical terms given by the grammar

$$i ::= 0 \mid 1 \mid x \mid i \times i \mid i + i,$$

where x is a first-order variable. The semantics is defined over a fixed structure $(\mathbb{R}, +, \times, \leq, 0, 1)$ of real arithmetic in the usual way. Relations definable by such formulae with additional real constants are called semi-algebraic.

Let M be an S-BSS machine and $n, t \in \mathbb{N}$ positive natural numbers. We denote by $L^n_t(M)$ ($L^n_{\leq t}(M)$, resp.) the set of strings $s \in \mathbb{R}^n$ accepted by M in time exactly (at most, resp.) t, and define $L^n(M) := L(M) \cap \mathbb{R}^n$. The following restricted fragment of $\exists$FO is enough to encode S-BSS computations.

Existential theory of the loose $[0,1]$***-guarded real arithmetic.*** Formulae of the *existential loose* $[0,1]$*-guarded real arithmetic* are defined as in (8), but without $i < i$ and replacing $\exists x \phi$ with $\exists x (0 \leq x \leq 1 \wedge \phi)$.

Lemma 4.3. *Given a deterministic or* $[0,1]$*-nondeterministic S-BSS machine M and positive $n, t \in \mathbb{N}$ it is possible to construct, in polynomial time, formulas ϕ and ψ of loose* $[0,1]$*-guarded real arithmetic, with free variables $x_1, \ldots, x_n$, that may use real constants used in M such that*

$$\{(s(x_1), \ldots, s(x_n)) \mid (\mathbb{R}, +, \times, \leq, (r)_{r \in \mathbb{R}}) \models_s \phi\} = L^n_t(M),$$
$$\{(s(x_1), \ldots, s(x_n)) \mid (\mathbb{R}, +, \times, \leq, (r)_{r \in \mathbb{R}}) \models_s \psi\} = L^n_{\leq t}(M).$$

Proof. For a given input of length n, the computation of M consists of t many configurations $\vec{c}_1, \ldots \vec{c}_t$ of M, where $\vec{c}_1$ and $\vec{c}_t$ are the initial configuration and a terminal configuration, respectively, and, for $1 \leq m < t$, $\vec{c}_{m+1}$ is a successor configuration of $\vec{c}_m$. Each configuration is a string of real numbers of length $O(t)$. We can use a similar technique as in the *right-to-left* direction of Theorem 3.3 and encode the contents of registers by pairs of real numbers from the unit interval $[0,1]$. In order to encode the computation, it suffices to encode the values of $O(t^2)$ registers; thus $O(t^2)$ variables suffice. We then construct a formula of existential loose $[0,1]$-guarded real arithmetic of size $O(t^2)$ that first existentially quantifies $O(t^2)$-many variables in order to guess the whole computation of M on the given input and then expresses, using perhaps at most polynomially many extra variables, that the computation is correct and accepting. We omit further details, for the encoding is done in a similar manner as in the *right-to-left* direction of Theorem 3.3. □

Given a deterministic S-BSS machine M, it is easy to see that the sets $L^n_t(M)$, for $n, t \in \mathbb{N}$, are disjoint. However, the same does not need to hold for nondeterministic machines, for the time it takes to accept an input string x might depend on the guessed value for the string x' (and there may be multiple accepting runs with different values for x'). This problem can be evaded for languages L that can be decided by a $[0,1]$-nondeterministic S-BSS machine N in time f, for some function $f : \mathbb{N} \to \mathbb{N}$. In this case $L^n(N) = L^n_{\leq f(n)}(N)$, for each $n \in \mathbb{N}$. Now since $L(M) = \bigcup_{n,t \in \mathbb{N}} L^n_t(M)$ and $L(N) = \bigcup_{n \in \mathbb{N}} L^n(N)$ where the unions are disjoint, we obtain the following characterisation.

Theorem 4.4. *Every language decidable by a) a deterministic S-BSS machine or b) a* $[0,1]$*-nondeterministic S-BSS machine in time t, for some $t : \mathbb{N} \to \mathbb{N}$, is a countable disjoint union of relations defined by existential loose* $[0,1]$*-guarded real arithmetic formulae that may use real constants from some finite set.*

The rest of this section is dedicated on proving the following theorem, which together with Theorem 4.4 implies Theorem 4.1.

Theorem 4.5. *Every relation defined by some existential loose* $[0,1]$*-guarded real arithmetic formula $\phi(x_1, ..., x_n)$ with real constants is closed in $\mathbb{R}^n$.*

Point-set topology. The proof of the theorem relies on some rudimentary notions and knowledge from point-set topology summarised in the following two lemmas (for basics of point-set topology see, e.g., the monograph [28]). In order to simplify the notation, for a topological space X, we use X to denote also the underlying set of the space. Likewise, in this section, we let $[0, 1]$ denote the topological space that has domain $[0, 1]$ and the metric of Euclidean distance.

Lemma 4.6. *Let X and Y be topological spaces, $f : X \to Y$ a continuous function, A and B closed sets in X, and C a closed set in Y. Then*

- *X, $A \cap B$, $A \cup B$, and $f^{-1}[C]$ are closed in X,*
- *the product $A \times C$ is closed in the product space $X \times Y$,*
- *if $Y \supseteq A$ is a subspace of X then A is closed in Y.*

Lemma 4.7. *Let X be a topological space, Y a compact topological space, A a closed set in the product space $X \times Y$, and f the projection function $X \times Y \to X$. Then the image $f[A]$ of A is closed in X.*

Proof of Theorem 4.5. We prove the following claim by induction on the structure of the formulae: Let $\vec{x}$ be a k-tuple of distinct variables and $\phi(\vec{x})$ an existential loose $[0,1]$-guarded real arithmetic formula with real constants, and its free variables in $\vec{x}$. The relation defined by $\phi(\vec{x})$ is closed in $\mathbb{R}^k$.

- Assume $\phi = t_1 \leq t_2$. Recall that $t_1(\vec{x})$ and $t_2(\vec{x})$ are multivariate polynomials. Define $g(\vec{x})$ as the multivariate polynomial $t_1(\vec{x}) - t_2(\vec{x})$ and consider the preimage $g^{-1}[(-\infty, 0]]$. Since $(-\infty, 0]$ is closed in $\mathbb{R}$ and

$g\colon \mathbb{R}^k \to \mathbb{R}$ is a continuous function, it follows that $g^{-1}[(\infty, 0]]$ is closed. Clearly $g^{-1}[(-\infty, 0]]$ is the relation defined by $\phi(\vec{x})$.

- The cases of disjunctions and conjunctions are clear, for the union and intersection of closed sets is closed.
- Assume $\phi = \exists y(0 \leq y \leq 1 \wedge \psi(\vec{x}, y))$. Let R_ψ be the relation defined by $\psi(\vec{x}, y)$, which by induction hypothesis is closed in $\mathbb{R}^{k+1}$. Define $R'_\psi := R_\psi \cap (\mathbb{R}^k \times [0, 1])$. Since $[0, 1]$ is closed in $\mathbb{R}$, it follows from Lemma 4.6 that R'_ψ is closed both in $\mathbb{R}^{k+1}$ and $\mathbb{R}^k \times [0, 1]$. Now let R^*_ψ be the projection of R'_ψ to its k first columns. Since R'_ψ is closed in $\mathbb{R}^k \times [0, 1]$, and $[0, 1]$ is a compact topological space, it follows from Lemma 4.7 that R^*_ψ is closed in $\mathbb{R}^k$. Clearly R^*_ψ is the relation defined by $\psi(\vec{x})$. □

5 Hierarchy of the complexity classes

The main result of this section is the separation of the complexity classes $\text{S-NP}_{[0,1]}$ and $\text{NP}_\mathbb{R}$. We have already done most of the work required for the separation as the result follows directly from the topological argument of Section 4.5 that more generally separates S-BSS computations from BSS computations. The characterisations of Section 3 then yield the separation of the related logics on $\mathbb{R}$-structures. We also give logical proofs implying that the obvious strengthenings of $\text{S-NP}_{[0,1]}$ coincide with $\text{NP}_\mathbb{R}$. Finally we study the restriction of $\text{S-NP}^0_{[0,1]}$ on Boolean inputs and establish that it coincides with a natural fragment of $\exists\mathbb{R}$.

5.1 Separation of $\text{S-NP}_{[0,1]}$ and $\text{NP}_\mathbb{R}$

We can now use Theorem 4.5 to prove the following:

Theorem 5.1. *The following separations hold:*

1. $\text{S-NP}^0_{[0,1]} < \text{NP}^0_\mathbb{R}$ *and* $\text{S-NP}_{[0,1]} < \text{NP}_\mathbb{R}$,
2. $\text{L-ESO}_{[0,1]}[+, \times, \leq, 0, 1] < \text{ESO}_\mathbb{R}[+, \times, \leq, 0, 1]$,
3. $\text{L-ESO}_{[0,1]}[+, \times, \leq, (r_{r\in\mathbb{R}})] < \text{ESO}_\mathbb{R}[+, \times, \leq, (r)_{r\in\mathbb{R}}]$.

Proof. We prove 1. by showing that there are languages in $\text{NP}^0_\mathbb{R}$ that are not in $\text{S-NP}_{[0,1]}$. The claims 2. and 3. then follow from the logical characterisations of Corollary 3.4.

Let L be a language in $\text{S-NP}_{[0,1]}$ and M an $\text{S-NP}_{[0,1]}$ S-BSS machine such that $L(M) = L$. Let p be a polynomial function that bounds the running time of M. Fix $n \in \mathbb{N}$. Now $L^n = L^n_{\leq p(n)}$. By Lemma 4.3 $L^n_{\leq p(n)}$, and hence L^n, is definable by an existential loose $[0, 1]$-guarded real arithmetic formula $\phi(x_1, ..., x_n)$ that uses real constants from M. By Theorem 4.5 L^n is a closed set in the product space $\mathbb{R}^n$, which is not true for all languages in $\text{NP}^0_\mathbb{R}$; for instance, a language P consisting of all finite strings of positive reals can be decided in $\text{NP}^0_\mathbb{R}$ (using branching), but P^n is not closed in $\mathbb{R}^n$. □

5.2 Robustness of $\text{NP}_\mathbb{R}$

We have just seen that $\text{S-NP}_{[0,1]}$ is a complexity class strictly below $\text{NP}_\mathbb{R}$. We now give purely logical proofs implying that the obvious strengthenings of $\text{S-NP}_{[0,1]}$ collapse to $\text{NP}_\mathbb{R}$. The proofs are based on the logical characterisations established in Corollary 3.4.

The first obvious question is: Are $\text{S-NP}_\mathbb{R}$ and $\text{S-NP}^0_\mathbb{R}$ strictly below $\text{NP}_\mathbb{R}$ and $\text{NP}^0_\mathbb{R}$? In logical terms this boils down to the expressivity of the logic $\text{L-ESO}_\mathbb{R}[+, \times, \leq, (r)_{r\in\mathbb{R}}]$. We answer to this question in the negative.

Proposition 5.2. $\text{L-ESO}_\mathbb{R}[+, \times, \leq, 0, 1] \equiv \text{ESO}_\mathbb{R}[+, \times, \leq]$ *and* $\text{L-ESO}_\mathbb{R}[+, \times, \leq, (r)_{r\in\mathbb{R}}] \equiv \text{ESO}_\mathbb{R}[+, \times, \leq, (r)_{r\in\mathbb{R}}]$.

Proof. The left-to-right direction is immediate as the constants 0 and 1 are definable in $\text{ESO}_\mathbb{R}[+, \times, \leq]$. For the converse direction, note that the numerical atom $\neg i \leq j$ is equivalent to the statement $j < i$. We show that $<$ is definable in $\text{L-ESO}_\mathbb{R}[+, \times, \leq, 0, 1]$. First note that every strictly positive real number $r \in \mathbb{R}$ can be expressed by a ratio of two real numbers $n, m \in \mathbb{R}$ such that $n, m \geq 1$. Moreover note that, for every such n and m, the ratio $n/m > 0$. It is easy to see that the following $\text{L-ESO}_\mathbb{R}[+, \times, \leq, 0, 1]$-formula

$$\exists r \exists n \exists m(1 \leq n \wedge 1 \leq m \wedge n = r \times m \wedge i + r = j),$$

where r, n, and m are 0-ary function variables, expresses that $i < j$. □

Theorem 2.4, Proposition 5.2, Corollary 3.4 together then yield the following:

Corollary 5.3. $\text{S-NP}_\mathbb{R} = \text{NP}_\mathbb{R}$ *and* $\text{S-NP}^0_\mathbb{R} = \text{NP}^0_\mathbb{R}$.

The second natural question is: Are $\text{NP}_{[0,1]}$ and $\text{NP}^0_{[0,1]}$ strictly below $\text{NP}_\mathbb{R}$ and $\text{NP}^0_\mathbb{R}$? Again, the answer is no. The proof of the following proposition follows directly from the observation that arbitrary real numbers can be encoded as ratios $x/(1-x)$, where $x \in [0, 1]$, using an additional marker for sign. It is crucial to note that with negated numerical atoms one can express that the denominators of such encodings are positive; in the loose fragment this is not possible. The encodings needed can be clearly expressed in $\text{ESO}_{[0,1]}[+, \times, \leq]$. We omit the proof.

Proposition 5.4. $\text{ESO}_{[0,1]}[+, \times, \leq, 0, 1] \equiv \text{ESO}_\mathbb{R}[+, \times, \leq, 0, 1]$ *and* $\text{ESO}_{[0,1]}[+, \times, \leq, (r)_{r\in\mathbb{R}}] \equiv \text{ESO}_\mathbb{R}[+, \times, \leq, (r)_{r\in\mathbb{R}}]$.

Hence Corollary 3.4 yields the following:

Corollary 5.5. $\text{NP}_{[0,1]} = \text{NP}_\mathbb{R}$ *and* $\text{NP}^0_{[0,1]} = \text{NP}^0_\mathbb{R}$.

Finally we consider a weakening of $\text{L-ESO}_\mathbb{R}[+, \times, \leq, 0, 1]$ by removing the constant 1 from the language. It turns out that this small weakening has profound implications to the expressivity of the logic when restricted to function-free vocabularies.

Proposition 5.6. *Let* $0 \in S \subseteq \mathbb{R}$. *Then* $\text{L-ESO}_S[+, \times, \leq] \equiv \text{FO}$ *with respect to* $\mathbb{R}$*-structures on function-free vocabularies.*

Proof. The direction FO $\leq$ L-ESO$_S[+,\times,\leq]$ is self-evident. We give a proof for the converse. Let $\mathfrak{A}$ be an $\mathbb{R}$-structure of a function-free vocabulary τ, $\phi \in$ L-ESO$_S[+,\times,\leq][\tau]$ a formula, and s an assignment for the first-order variables. Note that ϕ can be regarded also as a formula of L-ESO$_{\{0\}}[+,\times,\leq]$; we write ϕ_0 to denote this interpretation. Let $\phi_\top$ denote the FO-formula obtained from ϕ by removing the function quantifications in ϕ and replacing every numerical atom $i \leq j$ in ϕ with the formula $\exists x\, x = x$. Now note that there is a homomorphism from the first-order structure $(S,+,\times,\leq)$ to $(\{0\},+,\times,\leq)$, and consequently, $\mathfrak{A} \models_s \varphi \Leftrightarrow \mathfrak{A} \models_s \varphi_0$. Here we note that φ_0 implies φ since the second structure is a substructure of the first, and truth of existential formulae is preserved to extensions. Conversely, φ implies φ_0 because atoms $i \leq j$ appear only positively, and the truth of formulae with only positive literals are preserved to homomorphic images. Since in the evaluation of ϕ_0 every numerical term is evaluated to 0 it follows that $\mathfrak{A} \models_s \phi_0 \Leftrightarrow \mathfrak{A} \models_s \phi_\top$. □

5.3 Separate branching on Boolean inputs and the existential theory of the reals

It is known that on Boolean inputs $\mathsf{NP}^0_{\mathbb{R}}$ coincides with the complexity class $\exists\mathbb{R}$ (i.e., the class of problems polynomially reducible to the existential theory of the reals) [5, 26]. In this section we show an analogous result for S-$\mathsf{NP}^0_{[0,1]}$.

Definition 5.7. Define $\exists[0,1]^\leq$ to be the set of all languages $L \subseteq \{0,1\}^*$ for which there is a polynomial-time reduction f from $\{0,1\}^*$ into sentences of existential loose $[0,1]$-guarded real arithmetic such that $x \in L$ iff $(\mathbb{R},+,\times,\leq,0,1) \models f(x)$.

We show the following theorem:

Theorem 5.8. $\exists[0,1]^\leq = \mathrm{BP}(\text{S-}\mathsf{NP}^0_{[0,1]})$.

Proof. Note that the ***right-to-left*** direction of this theorem follows immediately from Lemma 4.3 by noting that the only real constants used by S-$\mathsf{NP}^0_{[0,1]}$ S-BSS machines M are 0 and 1, and that the Boolean inputs to M can be defined in $\exists[0,1]^\leq$ by using the constants 0 and 1.

Left-to-right. There exists a deterministic polynomial time Turing machine M that given an input string computes the corresponding sentence ϕ of existential loose $[0,1]$-guarded real arithmetic. Let p be the polynomial that bounds the running time of M. Without loss of generality we may assume that, for any given input i of length n, the formula computed by M from input i uses only variables $x_1,\ldots,x_{p(n)}$. Let M^* be a nondeterministic S-BSS machine that, for a given input i of length n, first guesses $p(n)$ many real numbers from the unit interval $[0,1]$ (these will correspond to the values of the variables $x_1,\ldots,x_{p(n)}$). Then M^* simulates the run of the deterministic polynomial time Turing machine M on input i. Let ϕ be the formula computed this way. Finally we can use M^* to check the matrix of ϕ using the values guessed for the variables $x_1,\ldots,x_{p(n)}$. We omit further details, for the evaluation of the matrix can done essentially in the same way as in the *left-to-right* direction of Theorem 3.3. □

6 Probabilistic team semantics

The purpose of this section is to characterise the descriptive complexity of probabilistic independence logic [10]. The formulae of this logic, and other logics that make use of dependency concepts involving quantities, are interpreted in probabilistic team semantics which generalises team semantics by adding weights on variable assignments. A finite model together with a probabilistic team can then be seen as a particular metafinite structure, and thus a natural approach to computational complexity comes from BSS machines.

Let D be a finite set of first-order variables, A a finite set, and X a finite set of assignments (i.e., a *team*) from D to A. A *probabilistic team* $\mathbb{X}$ is then defined as a function

$$\mathbb{X}\colon X \to [0,1]$$

such that $\sum_{s\in X}\mathbb{X}(s) = 1$. Also the empty function is considered a probabilistic team. We call D and A the variable domain and value domain of $\mathbb{X}$, respectively.

Probabilistic independence logic (FO($\perp\!\!\!\perp_c$)) is now defined as the extension of first-order logic with probabilistic independence atoms $\vec{y} \perp\!\!\!\perp_{\vec{x}} \vec{z}$ whose semantics is the standard semantics of conditional independence in probability distributions. Another probabilistic logic, FO($\approx$), is obtained by extending first-order logic with *marginal identity atoms* $\vec{x} \approx \vec{y}$ which state that the marginal distributions on $\vec{x}$ and $\vec{y}$ are identically distributed. The semantics for complex formulae are defined compositionally by generalising the team semantics of dependence logic to probabilistic teams. For details, not necessary in this paper, we refer the reader to [10]. In principle, the point is that formulae of probabilistic independence logic define properties of $(\mathfrak{A},\mathbb{X})$ where $\mathfrak{A}$ is a finite model and $\mathbb{X}$ a probabilistic team with value domain Dom($\mathfrak{A}$).

Example 6.1. Suppose we flip a coin. If we get heads, we roll two dice x and y. If we get tails, we roll only x and copy the same value for y. Repeating this procedure infinitely many times yields at the limit a probabilistic team (i.e., a joint probability distribution) over variables x and y satisfying

$$(x \perp\!\!\!\perp y \vee x = y) \wedge \forall z\, x \approx z.$$

By definition $\phi \vee \psi$ is true for a probabilistic team $\mathbb{X}$ if $\mathbb{X}$ is a mixture of two teams with respective properties ϕ and ψ (here independence and (row-wise) identity between x and y). By definition $\forall z\phi$ is true for a probabilistic team $\mathbb{X}$ if the extension of $\mathbb{X}$ with a uniform distribution for z has the property ϕ (here identity between marginal distributions on x and z).

We will now show that the descriptive complexity of probabilistic independence logic is exactly S-$\mathsf{NP}^0_{[0,1]}$. For this we need some background definitions and results.

Expressivity comparisons wrt. probabilistic team semantics. Fix a relational vocabulary τ. For a probabilistic team $\mathbb{X}$ with variable domain $\{x_1, \ldots, x_n\}$ and value domain A, the function $f_{\mathbb{X}} : A^n \to [0,1]$ is defined as the probability distribution such that $f_{\mathbb{X}}(s(\vec{x})) = \mathbb{X}(s)$ for all $s \in X$. For a formula $\phi \in \mathrm{FO}(\perp\!\!\!\perp_c)$ of vocabulary τ and with free variables $\{x_1, \ldots, x_n\}$, the class $\mathrm{Struc}(\phi)$ is defined as the class of $\mathbb{R}$-structures $\mathfrak{A}$ over $\tau \cup \{f\}$ such that $(\mathfrak{A} \upharpoonright \tau) \models_{\mathbb{X}} \phi$, where $f_{\mathbb{X}} = f^{\mathfrak{A}}$ and $\mathfrak{A} \upharpoonright \tau$ is the finite τ-structure underlying $\mathfrak{A}$.

Let $\mathcal{L}$ be any of the logics defined in Section 2. We write $\mathrm{FO}(\perp\!\!\!\perp_c) \leq \mathcal{L}$ if for every formula $\phi \in \mathrm{FO}(\perp\!\!\!\perp_c)$ of vocabulary τ there is a sentence $\psi \in \mathcal{L}$ of vocabulary $\tau \cup \{f\}$ such that $\mathrm{Struc}(\phi) = \mathrm{Struc}^{d[0,1]}(\psi)$. Vice versa, we write $\mathcal{L} \leq \mathrm{FO}(\perp\!\!\!\perp_c)$ if for every sentence $\psi \in \mathcal{L}$ of vocabulary $\tau \cup \{f\}$ there is a formula $\phi \in \mathrm{FO}(\perp\!\!\!\perp_c)$ of vocabulary τ such that $\mathrm{Struc}(\phi) = \mathrm{Struc}^{d[0,1]}(\psi)$.

Complexity characterisations wrt. probabilistic team semantics. Let $\mathrm{FO}(\perp\!\!\!\perp_c)$ be a logic with vocabulary τ and C a complexity class. Let $\mathcal{S}$ be an arbitrary class of $\mathbb{R}$-structures over $\tau \cup \{f\}$ that is closed under isomorphisms and where the interpretations of f are distributions. We write $\mathrm{enc}(\mathcal{S})$ for the set of encodings of structures in $\mathcal{S}$. Consider the following two conditions:

(i) $\mathrm{enc}(\mathcal{S}) = \{\mathrm{enc}(\mathfrak{A}) \mid \mathfrak{A} \in \mathrm{Struc}(\phi)\}$ for some $\phi \in \mathrm{FO}(\perp\!\!\!\perp_c)$.
(ii) $\mathrm{enc}(\mathcal{S}) \in C$.

If (i) implies (ii), we write $\mathrm{FO}(\perp\!\!\!\perp_c) \leq C$, and if the vice versa holds, we write $C \leq \mathrm{FO}(\perp\!\!\!\perp_c)$.

It is already known that probabilistic independence logic captures a variant of loose existential second-order logic in which function quantification ranges over distributions. This result was shown in two stages. First, it was proven in [10] that the logic $\mathrm{FO}(\perp\!\!\!\perp_c, \approx)$ is expressively equivalent to $\text{L-ESO}_{d[0,1]}[\mathrm{SUM}, \times, =]$.[2] Later, it was proven in [16] that marginal identity can be expressed using independence, that is, $\mathrm{FO}(\perp\!\!\!\perp_c, \approx)$ is expressively equivalent to $\mathrm{FO}(\perp\!\!\!\perp_c)$.[3]

Theorem 6.2 ([10, 16])**.** $\mathrm{FO}(\perp\!\!\!\perp_c) \equiv \text{L-ESO}_{d[0,1]}[\mathrm{SUM}, \times, =]$.

We will now improve this result by removing the condition that restricts function quantification to distributions. For this we utilize a normal form lemma from [10]. Observe that we restrict attention to $d[0,1]$-structures, that is, all function symbols from the underlying vocabulary are interpreted as distributions.

Lemma 6.3 ([10])**.** *For every* $\text{L-ESO}_{d[0,1]}[\mathrm{SUM}, \times, =]$*-formula* ϕ *there is an* $\text{L-ESO}_{d[0,1]}[\mathrm{SUM}, \times, =]$*-formula* ϕ^* *such that* $\mathrm{Struc}^{d[0,1]}\phi = \mathrm{Struc}^{d[0,1]}\phi^*$*, where* ϕ^* *is of the form* $\exists \vec{f} \forall \vec{x} \theta$*, where* θ *is quantifier-free and such that its second sort identity atoms are of the form* $f_i(\vec{u}, \vec{v}) = f_j(\vec{u}) \times f_k(\vec{v})$ *or* $f_i(\vec{u}) = \mathrm{SUM}_{\vec{v}} f_j(\vec{u}, \vec{v})$ *for distinct* f_i, f_j, f_k *such that at most one of them is not quantified.*

Lemma 6.4. $\text{L-ESO}_{d[0,1]}[\mathrm{SUM}, \times, =]$ $\equiv_{d[0,1]}$ $\text{L-ESO}_{d[0,1]}[+, \times, =]$ $\equiv_{d[0,1]}$ $\text{L-ESO}_{[0,1]}[+, \times, =, 0, 1]$.

Proof. We prove the claim in three steps, without relying on multiplication at any step. By Proposition 3.1 we may assume that the finite domain is enriched with a successor function S for tuples, its transitive derivatives $<, \leq$, and its minimal and maximal tuples $\vec{\min}$ and $\vec{\max}$ (of an appropriate arity), obtained by the lexicographic ordering induced from some linear ordering $\leq_{\mathrm{fin}}$. Additionally, we may assume a constant c on the finite domain.

Step 1: $\text{L-ESO}_{d[0,1]}[\mathrm{SUM}, \times, =] \leq_{d[0,1]} \text{L-ESO}_{d[0,1]}[+, \times, =]$. We may assume that any $\text{L-ESO}_{d[0,1]}[\mathrm{SUM}, \times, =]$ formula is of the form stated in Lemma 6.3. Thus it suffices to express in $\text{L-ESO}_{d[0,1]}[+, \times, =]$ each numerical identity of the form $f(\vec{u}) = \mathrm{SUM}_{\vec{x}} f'(\vec{u}, \vec{x})$. First, we quantify a $2m$-ary distribution variable g upon which we impose:

$$\begin{aligned} \forall \vec{x} \vec{y} \big[& g(\vec{x}, \vec{\min}) + g(\vec{x}, \vec{\min}) = f'(\vec{u}, \vec{x}) \wedge \\ & \big(\vec{y} < \vec{\max} \rightarrow \\ & g(S(\vec{y}), S(\vec{y})) + g(S(\vec{y}), S(\vec{y})) = g(S(\vec{y}), \vec{y}) + g(\vec{y}, \vec{y})\big) \wedge \\ & \big(S(\vec{y}) < \vec{x} \rightarrow \\ & g(\vec{x}, S(\vec{y})) + g(\vec{x}, S(\vec{y})) = g(\vec{x}, \vec{y})\big)\big]. \end{aligned} \tag{9}$$

The point is to calculate partial sums $\mathrm{SUM}_{\vec{x} \leq y} f'(\vec{u}, \vec{x})$ and store sufficiently small fractions of them in $g(\vec{y}, \vec{y})$. Suppose $\vec{y}$ is the nth tuple. Then

$$g(\vec{y}, \vec{y}) = \frac{1}{2^n}(f'(\vec{u}, \vec{\min}) + \ldots + f'(\vec{u}, \vec{y})),$$

and for $\vec{x} > \vec{y}$,

$$g(\vec{x}, \vec{y}) = \frac{1}{2^n} f'(\vec{u}, \vec{x}).$$

Consequently, the sum of all $g(\vec{x}, \vec{y})$ where $\vec{x} \geq \vec{y}$ is at most 1. By allocating the remaining weights to $(\vec{x}, \vec{y})$ such that $\vec{x} < \vec{y}$, it follows that g is a distribution.

Furthermore, we quantify a $2m$-ary distribution variable h satisfying:

$$\begin{aligned} \forall \vec{x} [& h(\vec{\min}) + h(\vec{\min}) = f(\vec{u}) \wedge \\ & \vec{x} < \vec{\max} \rightarrow h(S(\vec{x})) + h(S(\vec{x})) = h(\vec{x})]. \end{aligned}$$

It follows that $h(\vec{y}) = \frac{1}{2^n} f(\vec{u})$. Consequently, $g(\vec{\max}, \vec{\max}) = h(\vec{\max})$ if and only if $f(\vec{u}) = \mathrm{SUM}_{\vec{x}} f'(\vec{u}, \vec{x})$. Note that h is not a distribution since the weights do not add up to 1. However, we may increment the arity of h by one and replace $h(\vec{x})$ above with $h(\vec{x}, c)$. Then h is a distribution if the remaining weights are pushed to $h(\vec{x}, y)$, where $y \neq c$. This concludes the proof of Step 1.

[2] In [10] equi-expressivity with $\mathrm{ESO}_{d[0,1]}[\mathrm{SUM}, \times, =]$ is erroneously stated; the results in the paper actually entail equi-expressivity with $\text{L-ESO}_{d[0,1]}[\mathrm{SUM}, \times, =]$.

[3] In fact, $\mathrm{FO}(\perp\!\!\!\perp_c)$ is expressively equivalent to $\mathrm{FO}(\perp\!\!\!\perp)$ which is the extension of first-order logic with *marginal independence atoms* $\vec{x} \perp\!\!\!\perp \vec{y}$, the semantics of which is the standard semantics of marginal independence in probability distributions [16].

Step 2: We show a stronger claim: L-ESO$_{d[0,1]}[+,\times,=] \leq$ L-ESO$_{[0,1]}[+,\times,=,0,1]$. For this, it suffices to show how to express in L-ESO$_{[0,1]}[+,=,0,1]$ that a function f is a distribution. The following formula expresses just that:

$$\exists g\big(g(\vec{\min}) = f(\vec{\min}) \wedge \forall \vec{x}(\vec{x} < \vec{\max} \rightarrow g(S(\vec{x})) = g(\vec{x}) + f(S(\vec{x}))) \wedge g(\vec{\max}) = 1\big).$$

Step 3: We show a stronger claim: L-ESO$_{[0,1]}[+,\times,=,0,1]$ $\leq_{[0,1]}$ L-ESO$_{d[0,1]}[\mathrm{SUM},\times,=]$. Suppose ϕ is some formula in L-ESO$_{[0,1]}[+,\times,=,0,1]$. Let k be the maximal arity of any function variable/symbol appearing in ϕ, and suppose n is the size of the finite domain; the total sum of the weights of a function is thus at most n^k. We now show how to obtain from ϕ an equivalent formula in L-ESO$_{d[0,1]}[\mathrm{SUM},\times,=]$; the idea is to scale all function weights by $1/n^k$. We have two cases:
Function variables. If f is an m-ary quantified function variable, we replace it with an $(m+1)$-ary quantified distribution variable d_f satisfying

$$\forall \vec{x} \exists d' \forall \vec{y}\, d'(\vec{y}, c) = d_f(\vec{x}, c),$$

where d' is a $(k+1)$-ary distribution variable. Now $n^k d_f(\vec{x}, c) \leq 1$ because d' is a distribution, and thus $d_f(\vec{x}, c) \leq \frac{1}{n^k}$.
Function symbols. Suppose $f(\vec{x})$ is a function term which appears as a term or subterm in ϕ, and f is a function symbol from the underlying vocabulary. We quantify a $(k+1)$-ary distribution variable $d_{f(\vec{x})}$ satisfying

$$\forall \vec{x}(\mathrm{SUM}_{\vec{y}} d_{f(\vec{x})}(\vec{y}, c) = f(\vec{x}) \wedge \forall \vec{y} \vec{z} d_{f(\vec{x})}(\vec{y}, c) = d_{f(\vec{x})}(\vec{z}, c)).$$

It follows that $d_{f(\vec{x})}(\vec{x}, c) = \frac{1}{n^k} f(\vec{x})$. Since $f(\vec{x}) \leq 1$, we may define $d_{f(\vec{x})}$ as a distribution.

Observe now that each numerical atom appearing in ϕ is an identity between two multivariate polynomials over function terms. Without loss of generality all the constituent monomials in these atoms are of a fixed degree D and have coefficient one; note that each monomial with degree less than D can be appended in L-ESO$_{[0,1]}[+,\times,=,0,1]$ with a quantified nullary function n taking value 1. We now replace in each numerical atom $i = j$ function terms $f(\vec{x})$ with $d_f(\vec{x}, c)$ or $d_{f(\vec{x})}(\vec{x}, c)$, depending on whether f is a function variable or a function symbol. Thus we represent $i = j$ in L-ESO$_{d[0,1]}[\mathrm{SUM},\times,=]$ as $\frac{i}{n^{Dk}} = \frac{j}{n^{Dk}}$, wherefore not only its truth value, but also that of ϕ, is preserved in the transformation. □

By combining Corollary 3.4.3, Theorem 6.2, and Lemma 6.4, we finally obtain the following result.

Theorem 6.5. $\mathrm{FO}(\perp\!\!\!\perp_c) \equiv \text{S-NP}^0_{[0,1]}$.

7 Concluding remarks

Applications of logic in AI and advanced data management require probabilistic interpretations, a role that is well fulfilled by probabilistic team semantics. On the other hand, in the theory of computation and automated reasoning, computation and logics over the reals are well established with solid foundations. In this paper we have provided bridges between the two worlds. We introduced a novel variant of BSS machines and provided a logical and topological characterisation of its computational power. In addition, we determined the expressivity of probabilistic independence logic with respect to the BSS model of computation.

There are many interesting directions of future research. One is to consider the additive fragment of BSS computation. Restricted to Boolean inputs it is known that, if unrestricted use of machine constants is allowed, the additive $\mathsf{NP}_{\mathbb{R}}$ branching on equality collapses to NP and branching on inequality captures NP/*poly* [21]. What can we say about the additive fragment of S-BSS computation? Another direction is to devise logics that characterise other important complexity classes over S-BSS machines. Grädel and Meer [15] established a characterisation of polynomial time on ranked $\mathbb{R}$-structures using a variant of least fixed point logic. In the setting of team semantics and classical computation, Galliani and Hella [12] showed that the so-called *inclusion logic* characterises polynomial time on ordered structures. Can we extend the applicability of these results to the realms of S-BSS computation and probabilistic team semantics? Finally, we would like to devise natural complete problems for the complexity classes defined by S-BSS machines. In particular, we would like to obtain a natural complete problem for $\exists[0,1]^{\leq}$; a weakening of the art gallery problem is one promising candidate.

We conclude with a few open problems:

- Is $\exists[0,1]^{\leq}$ strictly included in $\exists\mathbb{R}$? A positive answer would be a major breakthrough, as it would separate NP from PSPACE.
- We know that $\mathsf{NP} \leq \exists[0,1]^{\leq} \leq \exists\mathbb{R} \leq \mathsf{PSPACE}$. Can we establish a better upper bound for $\exists[0,1]^{\leq}$? In particular, is $\exists[0,1]^{\leq}$ contained in the polynomial hierarchy?
- We established that S-BSS computable languages are included in the class of BSS computable languages that are countable disjoint unions of closed sets. Does the converse hold?

Acknowledgments

The first and second authors were supported by the Academy of Finland grant 308712. The third and fourth authors were supported by the Research Foundation Flanders grant G0G6516N. The third author was partially supported by the National Natural Science Foundation of China under grant 61972455, and the fourth author was an international research fellow of the Japan Society for the Promotion of Science, Postdoctoral Fellowships for Research in Japan (Standard).

References

[1] Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. 2018. The art gallery problem is $\exists\,\mathbb{R}$-complete. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018,*

Los Angeles, CA, USA, June 25-29, 2018. 65–73. https://doi.org/10.1145/3188745.3188868

[2] Michael Benedikt, Martin Grohe, Leonid Libkin, and Luc Segoufin. 2003. Reachability and connectivity queries in constraint databases. *J. Comput. System Sci.* 66, 1 (2003), 169 – 206. https://doi.org/10.1016/S0022-0000(02)00034-X Special Issue on PODS 2000.

[3] Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. 1997. *Complexity and Real Computation*. Springer-Verlag, Berlin, Heidelberg.

[4] Lenore Blum, Mike Shub, and Steve Smale. 1989. On a theory of computation and complexity over the real numbers: NP- completeness, recursive functions and universal machines. *Bull. Amer. Math. Soc. (N.S.)* 21, 1 (07 1989), 1–46. https://projecteuclid.org:443/euclid.bams/1183555121

[5] Peter Bürgisser and Felipe Cucker. 2006. Counting complexity classes for numeric computations II: Algebraic and semialgebraic sets. *J. Complexity* 22, 2 (2006), 147–191. https://doi.org/10.1016/j.jco.2005.11.001

[6] John F. Canny. 1988. Some Algebraic and Geometric Computations in PSPACE. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*. 460–467. https://doi.org/10.1145/62212.62257

[7] Jukka Corander, Antti Hyttinen, Juha Kontinen, Johan Pensar, and Jouko Väänänen. 2019. A logical approach to context-specific independence. *Ann. Pure Appl. Logic* 170, 9 (2019), 975–992. https://doi.org/10.1016/j.apal.2019.04.004

[8] Felipe Cucker and Klaus Meer. 1999. Logics Which Capture Complexity Classes Over The Reals. *J. Symb. Log.* 64, 1 (1999), 363–390. https://doi.org/10.2307/2586770

[9] Arnaud Durand, Miika Hannula, Juha Kontinen, Arne Meier, and Jonni Virtema. 2018. Approximation and dependence via multiteam semantics. *Ann. Math. Artif. Intell.* 83, 3-4 (2018), 297–320. https://doi.org/10.1007/s10472-017-9568-4

[10] Arnaud Durand, Miika Hannula, Juha Kontinen, Arne Meier, and Jonni Virtema. 2018. Probabilistic Team Semantics. In *Foundations of Information and Knowledge Systems - 10th International Symposium, FoIKS 2018, Budapest, Hungary, May 14-18, 2018, Proceedings*. 186–206. https://doi.org/10.1007/978-3-319-90050-6_11

[11] Pietro Galliani. 2008. Game Values and Equilibria for Undetermined Sentences of Dependence Logic. (2008). MSc Thesis. ILLC Publications, MoL–2008–08.

[12] Pietro Galliani and Lauri Hella. 2013. Inclusion Logic and Fixed Point Logic. In *Computer Science Logic 2013 (CSL 2013) (Leibniz International Proceedings in Informatics (LIPIcs))*, Simona Ronchi Della Rocca (Ed.), Vol. 23. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 281–295. https://doi.org/10.4230/LIPIcs.CSL.2013.281

[13] Erich Grädel and Yuri Gurevich. 1998. Metafinite Model Theory. *Inf. Comput.* 140, 1 (1998), 26–81. https://doi.org/10.1006/inco.1997.2675

[14] Erich Grädel and Stephan Kreutzer. 1999. Descriptive Complexity Theory for Constraint Databases. In *Computer Science Logic, 13th International Workshop, CSL '99, 8th Annual Conference of the EACSL, Madrid, Spain, September 20-25, 1999, Proceedings*. 67–81. https://doi.org/10.1007/3-540-48168-0_6

[15] Erich Grädel and Klaus Meer. 1995. Descriptive complexity theory over the real numbers. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA*. 315–324. https://doi.org/10.1145/225058.225151

[16] Miika Hannula, Åsa Hirvonen, Juha Kontinen, Vadim Kulikov, and Jonni Virtema. 2019. Facets of Distribution Identities in Probabilistic Team Semantics. In *JELIA (Lecture Notes in Computer Science)*, Vol. 11468. Springer, 304–320.

[17] Miika Hannula and Juha Kontinen. 2016. A finite axiomatization of conditional independence and inclusion dependencies. *Inf. Comput.* 249 (2016), 121–137. https://doi.org/10.1016/j.ic.2016.04.001

[18] Uffe Flarup Hansen and Klaus Meer. 2006. Two logical hierarchies of optimization problems over the real numbers. *Math. Log. Q.* 52, 1 (2006), 37–50. https://doi.org/10.1002/malq.200510021

[19] Tapani Hyttinen, Gianluca Paolini, and Jouko Väänänen. 2017. A Logic for Arguing About Probabilities in Measure Teams. *Arch. Math. Logic* 56, 5-6 (2017), 475–489. https://doi.org/10.1007/s00153-017-0535-x

[20] Paris C. Kanellakis, Gabriel M. Kuper, and Peter Z. Revesz. 1995. Constraint Query Languages. *J. Comput. Syst. Sci.* 51, 1 (1995), 26–52. https://doi.org/10.1006/jcss.1995.1051

[21] Pascal Koiran. 1994. Computing over the Reals with Addition and Order. *Theor. Comput. Sci.* 133, 1 (1994), 35–47. https://doi.org/10.1016/0304-3975(93)00063-B

[22] Andreas Krebs, Arne Meier, Jonni Virtema, and Martin Zimmermann. 2018. Team Semantics for the Specification and Verification of Hyperproperties. In *MFCS (LIPIcs)*, Vol. 117. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 10:1–10:16.

[23] Stephan Kreutzer. 2000. Fixed-Point Query Languages for Linear Constraint Databases. In *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, May 15-17, 2000, Dallas, Texas, USA*. 116–125. https://doi.org/10.1145/335168.335214

[24] Klaus Meer. 2000. Counting problems over the reals. *Theor. Comput. Sci.* 242, 1-2 (2000), 41–58. https://doi.org/10.1016/S0304-3975(98)00190-X

[25] Marcus Schaefer. 2009. Complexity of Some Geometric and Topological Problems. In *Graph Drawing, 17th International Symposium, GD 2009, Chicago, IL, USA, September 22-25, 2009. Revised Papers*. 334–344. https://doi.org/10.1007/978-3-642-11805-0_32

[26] Marcus Schaefer and Daniel Stefankovic. 2017. Fixed Points, Nash Equilibria, and the Existential Theory of the Reals. *Theory Comput. Syst.* 60, 2 (2017), 172–193. https://doi.org/10.1007/s00224-015-9662-0

[27] Jouko Väänänen. 2007. *Dependence Logic*. Cambridge University Press.

[28] S. Willard. 2004. *General Topology*. Dover Publications. https://books.google.co.jp/books?id=-o8xJQ7Ag2cC